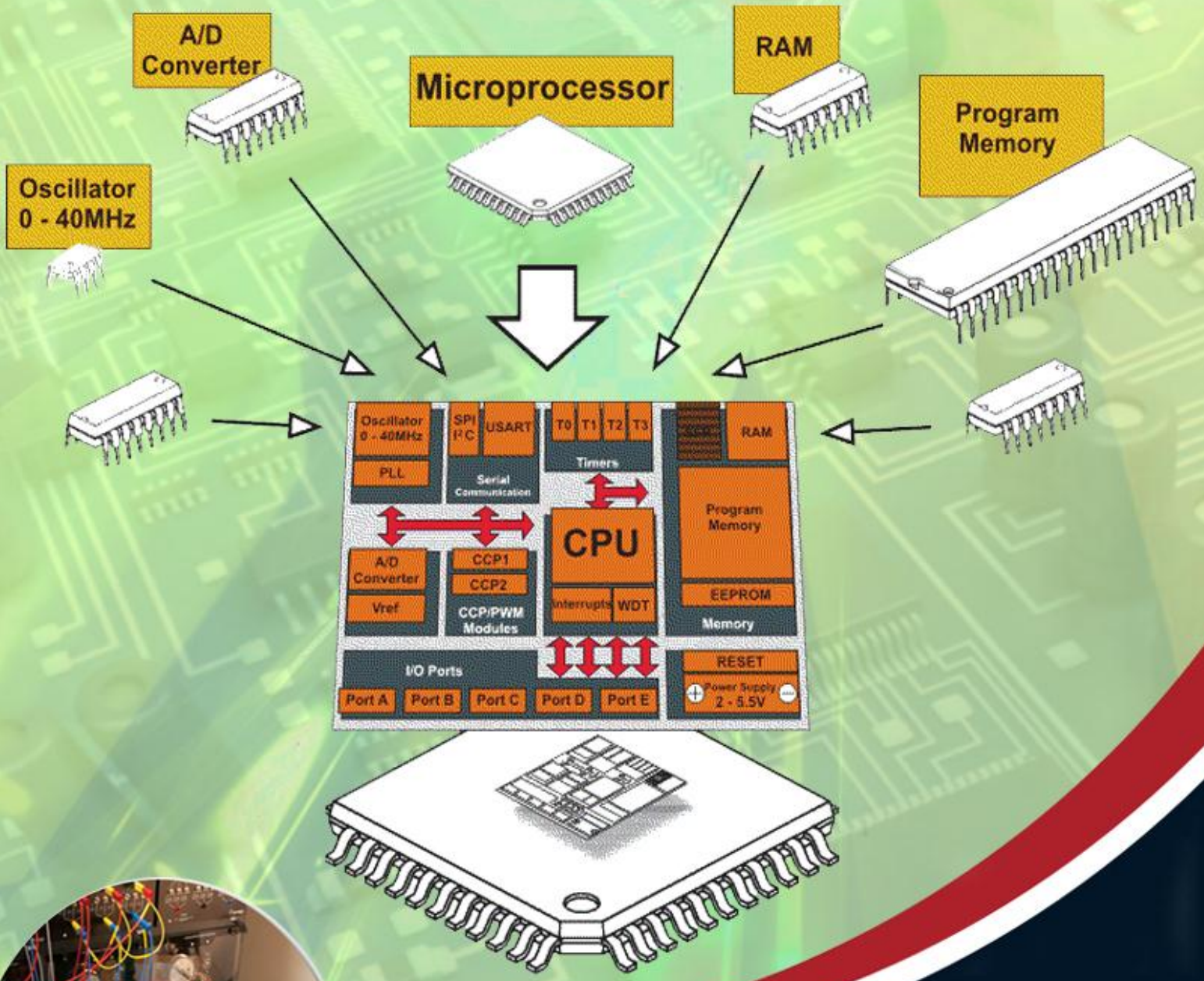
microcontroller





www.az-ex.net
twitter.com/azex2013
www.facebook.com/alazharexhibition
www.facebook.com/groups/azex.2013

مقدمة

بفضل الله وتوفيقه أتممنا تجهيز وتحضير هذه المادة العلمية للمساهمة في العملية التعليمية بكلية الهندسة جامعة الأزهر، وذلك من خلال تقوية الجانب العملي لدى الطلاب لتكون لديهم القدرة على المنافسة في سوق العمل، ولمساعدتهم على إقامة المشاريع المختلفة سواءً للمشاركة في معرض الأزهر للتطبيقات الهندسية أو غيره.

وقام بإعداد هذه المادة العلمية مجموعة من طلاب الفرقة الثالثة بكلية الهندسة بجامعة الأزهر، وقد حاول كل منا إضافة كل ما لديه من خبرة في هذه المادة العلمية، وذلك بالإضافة إلى مصادر أخرى مثل بعض المواقع والمنتديات على الإنترنت.

وقد راعينا أن تكون المادة العلمية موحدة الأسلوب، بطريقة تتيح للطالب أن يكون ملما بالأساسيات ليتمكن الطالب من دراستها بسهولة ويسر.

وهذا العمل بما يحتويه من شرح نظري أو مشاريع عملية يمكن توزيعها ونسخها دون الرجوع إلينا.

و أخيرا وليس آخراً يتقدم فريق الإعداد بخالص الشكر والتقدير لكل من ساعدنا في إخراج هذا العمل على الصورة المرجوة، سائلين المولى عز وجل أن يوفقنا ويوفق جميع طلاب العلم وأن يجعلنا سببا في نفعهم.

لجنة الدعم الفني

AZ-EX 2013

فريق الإعداد

أحمد محمود الدابولي
eldaboly_2010@yahoo.com

أمال محمود الرزاز
y_m729@yahoo.com

أنس إبراهيم محمد
anas-ebrahim@hotmail.com

أنس خليل خليل محمد
anas_khalil@live.com

بلال عصام الديواني
belal.eldiwany@hotmail.com

منة الله محمد الحسينى المداح
flying_ butterfly91@yahoo.com

وسام مصطفى السيد
soma_mostafa2020@hotmail.com

Table of contents:

Chapter (1)" Introduction" ---------- Page 6

Chapter (2)"Mikro C" ---------- Page 25

Chapter (3)" Input & output devices" ---------- Page 50

Chapter (4)" Projects" ---------- Page 58

Chapter (5)" 7-Segment" ---------- Page 105

Chapter (6)" LCD" ---------- Page 128

Chapter (7)"DC motor" ---------- Page 144

Chapter (8)" ADC" ---------- Page 176

Chapter (9)" Keypad" ---------- Page 182

Chapter (10)" Stepper" ---------- Page 196

Chapter (11)" Serial communication" ---------- Page 214

Table of components:

1-PIC16F877a

2- RESISTORS WITH DIFFERENT VALUES

3-LM016 (LCD DISPLAY)

4-PUSH PUTTONS

5-TRANSISTORS (BC547 OR 548 & 2N2222 &BC337 & TIP122 & TIP127)

6-DIODES (1N4007 OR 1N4001)

7-LEDS WITH DIFFERENT COLOURS

8-VARIABLE RESISTOR

9-ICS (LM35 & ULN2803 & 4514<DECODER>)

10-LDR

11-BUZZER

12-RELAYS 5V

13-KEYPAD

14-SEVEN SEGMENT

15-PIN HEADER

16-REGULATOR (7805)

17-Capacitor 10µF

18-Capacitor 20pF

19-Crystal 4MHz

20-MAX232

Chapter (1)

Introduction



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

Microcontroller

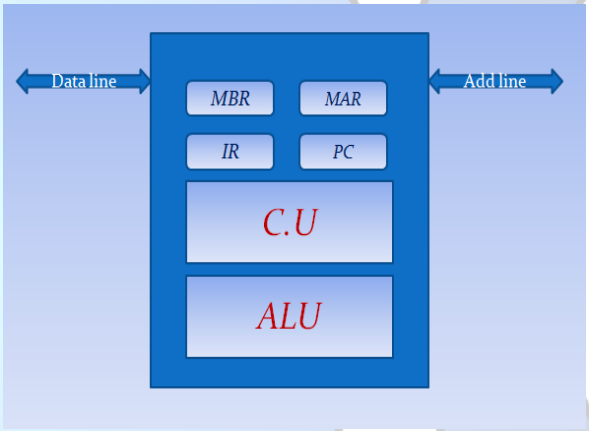
الميكروبروسيسورMicroprocessor

عبارة عن وحدة معالجة للبيانات كامله ومدمجة أو مجمعه فى IC واحد وعادة ما يطلق عليها (CPU) أو وحدة المعالجة المركزية وهو يقوم بفهم مجموعه من الاوامر ويقوم بتنفيذها ليقوم بوظيفه معينه.

يتألف المايكروبروسيسورمن 3 وحدات رئيسية:

1 ALU (arithmetic & logic unit):

وهى خاصة بعمليات ال processing سواء كانت عمليات حسابية (جمع / طرح ...) أو منطقية (XOR) .



2 REGISTERS:

وهى خاصة بتخزين البيانات الاتية من ال DATA LINES وتحويلها الى ALU لتقوم بمعالجتها ثم تخزينها مرة اخرى فى ال REGISTER .

3 CU (control unit)

مكونات المعالج الدقيق CPU

تتألف ال CPU من 3 وحدات رئيسية هما :

1_ oscillator :

لكى يعمل CPU بكفاءة عالية لابد ان تكون كل العمليات التى تمت معالجتها متزامنة وتسير وفقا لساعه داخلية عالية فى الدقة عن طريق استخدام oscillator وهو عبارة عن ساعة داخلية لاخراج اشارة زمنية ثابته (clock) هذة الساعة عبارة عن وحدة يتم تركيبها تسمى Crystal يتم تصنيعها من مقاومة ومكثف وتسمى بساعه انتظام).(system clock

2_ Memory :

يتم فيها تخزين البيانات التى تستخدم بواسطة CPU وهى نوعان :

_A RAM (read & write memory) :

يستطيع MP ان يكتب البيانات على هذا النوع من الذاكرة وايضا يقرأ البيانات من هذة الذاكرة والبيانات على هذة الذاكرة يمكن تغيرها فى اى وقت .

_B ROM (read only memory) :

يستطيع MP ان يقرأ البيانات من على ROM ولكنه لا يستطيع ان يكتب عليه اى بيانات والبيانات داخل ROM تسمى firmware يتم تخزينها اثناء عملية التصنيع ولا يمكن تعديلها عن طريق المستخدم .

_3 o/p ports :

تعمل interface بين ال processor والعالم الخارجى حيث ان المدخلات غالبا تكون عبارة عن مجموعه من الازرار والمفاتيح اما ال outputs تكون على شاشة وحدة عرض .

_4 timers :

مؤقت يحدد عليه زمن معين بعد مرورة يقوم ال processor بأداء وظيفة معينه.

_5 communication interface

_6 buses :

كل مكونات ال micro computer تتصل ببعضها عن طريق كابلات او توصيلات تسمى buses

Example:

برنامج التكييف مثلا يحتاج الى:

1_ read set point (user desired temperature)

2_ read room temperature with sensor

3_ if room temp. > Set point ⟶ Turn compressor on

4_ if room temp. < Set point ⟶ Turn compressor off

نلاحظ ان ال set point تخزن فى ال ram كمتغير X لانى قريت وكتبت

وايضا ال .room temp تخزن فى ال ROM كمتغير T لانى قريت وكتبت

T>X

T<X

مشاكل الميكروبروسسور:

1_ لا يمكنه التعامل مع اشارة ANALOG ولكن digital فقط والحل لهذة المشكلة استخدم دائرة ADC للتحويل من analog الى ديجيتال ليتعرف الميكروبرسسور ع الاشارة .

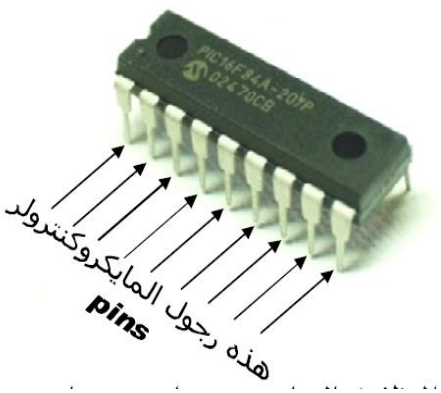
2_ لا يمكن ربطه بالكومبيوتر لان عملية ربط الكومبيوتر بأى شئ خارجى تتم عن طريق ال ports التى بداخله سواء USB PORT أو SERIAL PORT وال serial port يتم استقبال البيانات بالتتابع اما ال microprocessor بيستقبل الداتا مرة واحدة بمعنى انه بيستقبلها parallel ولحل هذة المشكلة نستخدم serial interface لتحويل الداتا ال parallel الخارجة من الميكروبروسسور الى داتا serial يفهمها الكومبيوتر أو العكس .

3_ لا يمكن ربطة بالنت مباشرة وذلك لانه عند إرسال بيانات الى النت فانه يتم اضافة عليها parity check فلو مثلا ارسلنا اشارة مكونة من 00000011 فانظر الى عدد الواحد فى الرسالة هل هو زوجى أم فردى فإذا كان زوجى نضيف للرسالة خانة زيادة فيها zero فترسل هكذا 000000011 واذا كان عدد الواحد فردى نضيف للرسالة خانة زيادة فيها one فينظر ال interface على ال parity check هل هو واحد ام صفر فإذا كان صفر ينظر للرساله هل عدد الواحد زوجى ام لا فإذا كان زوجى يقوم بارسال الرساله للنت بعد حذف الاضافات لو كان عدد ال ones فردى يقوم بارسال رسالة ان البيانات المبعوثة خاطئة ولحل هذة المشكلة استخدم دائرة ادخل عليها البيانات المرسلة تقوم بارسال البيانات الى النت بعد إضافة ال parity check ويسمى هذا ال interface بال Ethernet .

4_ لا يحتوى الميكروبروسسور على internal memory ولكنه يحتاج الى hard disk وايضا الى RAM .

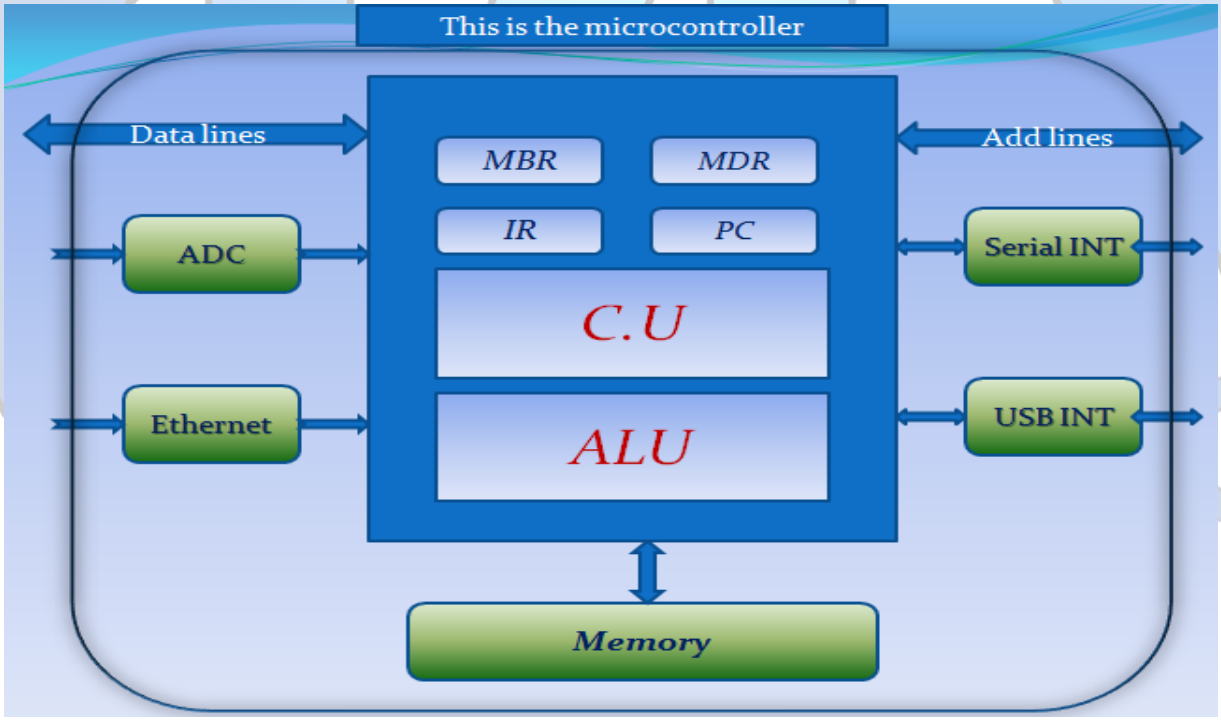
طب دلوقت عدد ال Components كتير ممكن 10 ICS فبالتالى حجم كبير يعنى خطأ اكثر ومن هنا جاءت فكرة ال :Micro Controller

المايكروكنترولر MCU



عبارة عن كل المكونات السابقة ولكنها متجمعه فى IC واحده SINGLE CHIP يمكن التحكم فى خصائصة ومكوناته باستخدام SOFTWARE .

مكونات المايكروكنترولر من الداخل ما هو إلا كمبيوتر صغير Minicomputer حيث يتكون من وحدة معالجة CPUوكذلك ذاكرة عشوائية RAM و ذاكرة من النوع ROM بالإضافة إلى وحدة تخزين يوضع عليها البرامج والبيانات (كما في الكمبيوتر الشخصي العادي) بالإضافة إلى وحدة الإدخال والإخراج حيث يكون الإدخال والإخراج عن طريق رجول المايكروكنترولر (Pins) وأقصد بالرجول أطرافه حيث يخرج المايكروكنترولر الإشارات الكهربية وكذلك يستقبلها, كل ذلك تتحكم فيه أنت كما تريد من خلال عملية البرمجة.



مقارنة بين الميكروبروسيسور والميكروكنترولر :

مميزات ال Microcontroller :

1_ عدد المكونات اقل من المكونات المطلوبة فى ال Microcomputer ..

2_ رخيص

3_ نسبة الخطأ اقل وسهل الاستخدام لقلة مكوناته

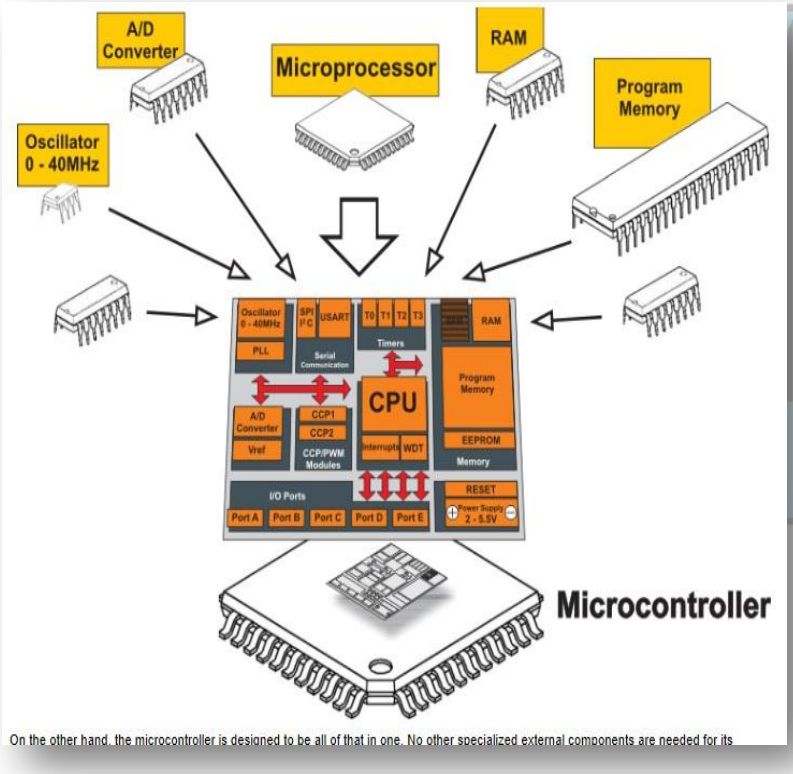
مميزات ال microcomputer :

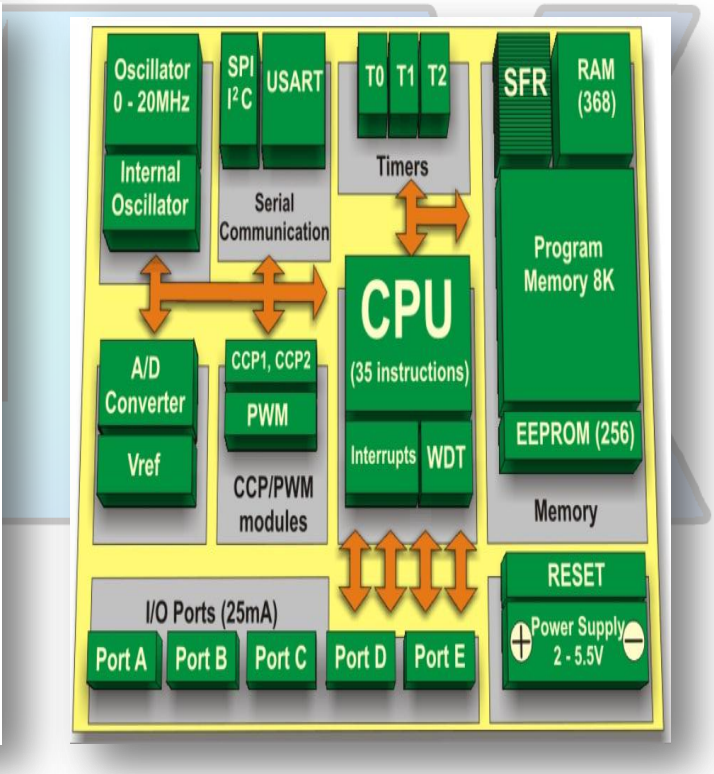
سهوله تغيير مكوناته لكن فى الميكروكنترولر لا يمكن تغيير الذاكرة او اى شىء ومن هنا يمكن تحديث ال microcomputer باجهزة ذو مواصفات احسن .

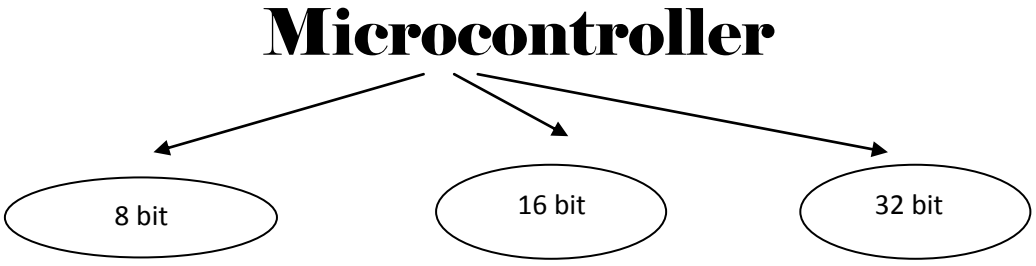
الخلاصة :

يستحب استعمال ال microprocessor فى اجهزة الحاسب لامكانية التغيير فيها بمواصفات احسن ويستحب استخدام الميكروكنترولر فى باقى التطبيقات تقريبا لانها ذو برنامج ثابت مثل التكييف لا يحتاج الى تحديث .

حتى فى الموبيل يفضل استخدام الميكروكنترولر ليكون حجم الموبيل صغير .







ال 8 bit

كل register يتكون من 8 خانات يقدر يخزن فى اى register

255= 2^8 -1 لو أراد جمع رقمين كل منهم 8 bit يقسم العمليه على كذا مرحله بمعنى أنه لو عندى رقمين كل منهم 16 bit تتم عمليه الجمع على 3 مراحل نظرا لوجود Carry .

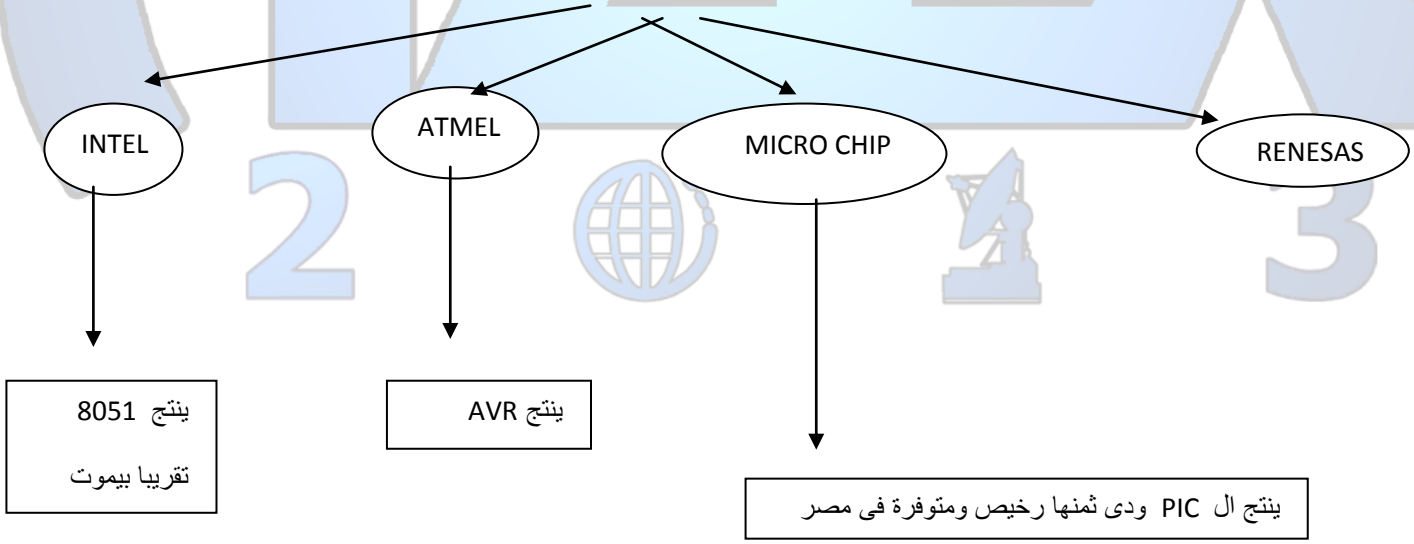
لاحظ :

اكثر الانواع من حيث البيع هو ال 8 byte لانه أرخصهم فهيكون نسبه استعمال هذا النوع فى السوق حوالى 3/2 والنوعين الاخريين 3/1 حيث ان معظم تطبيقات الميكروكنترولر لا يدخل فيها حسابات فلا احتاج الى عدد bits اكتر .

Types of microcontroller

كل شركة قامت بتصنيع الميكروكونترولر فإنها أطلقت عليه اسم من المحتمل أن يكون اسم الشركة أم لآ .

يبيع ال 8 bite الكثير من شركات ال semiconductor امثال :



ليه يفضل استخدام ال PIC عن ال AVR ؟؟؟؟؟

لان قدرة تحمل ال PIC لل .temp وكذلك الخطأ فى عكس ال polarity لل powerبيكون اكتر من ال AVR .

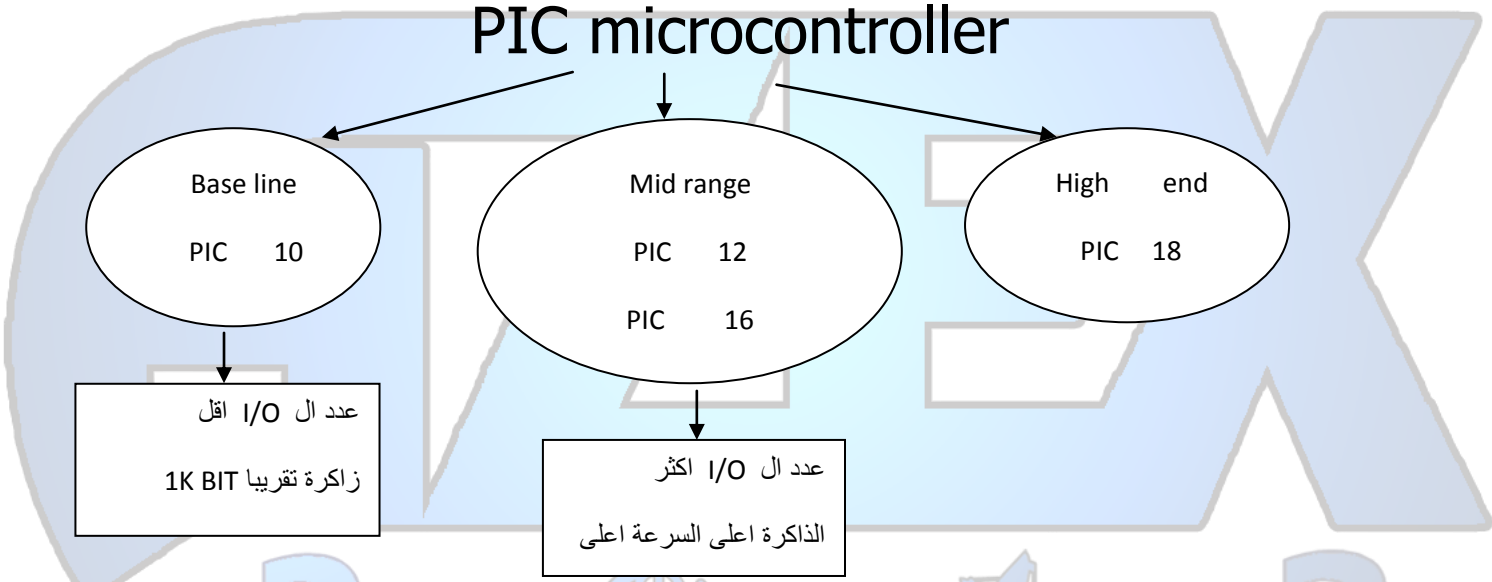
الخلاصة ان ال stability لل PIC اكبر .

احيانا نواجه مشكله ان الناس اللى عملوا روبوت بيشتغل بميكروكنترولر فشلوا واللى عملوا روبوت ب plc نجحوا ايه السبب !!!!!!!!!!!!!!!؟؟؟؟؟؟؟؟؟

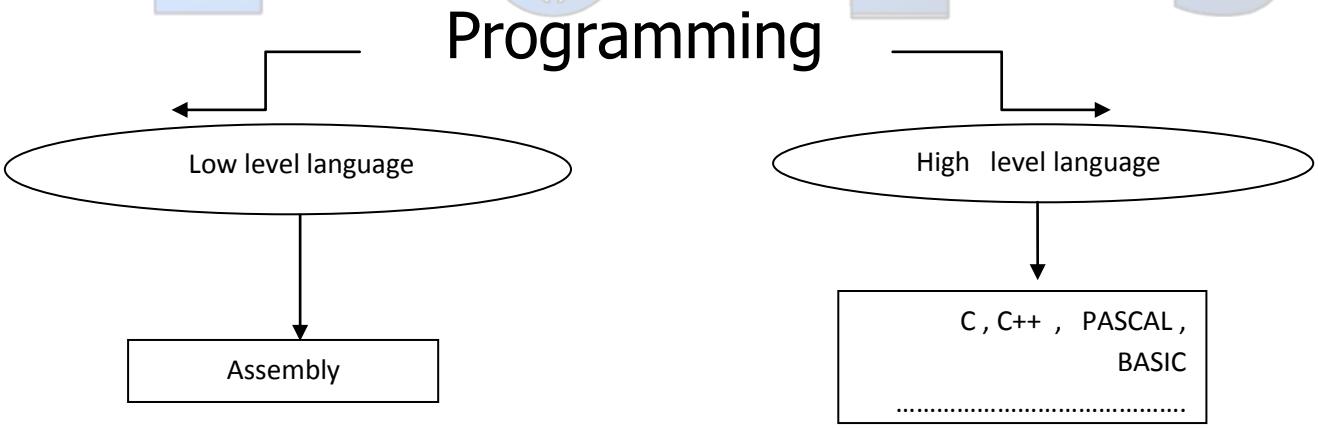
ال plc اصلا ميكروكنترولر الفرق فقط ان ال hardware جاهز يعنى اكتر مشكله فى الروبوت انه بيشتغل ببطارية لو مش مغذى الميكرو بب power نظيفة الجهاز هيخرف ال plc مش باظ لان ال plc له power مستقل عن ال power اللى بيغذى الموتور عشان اعمل روبوت بميكرو ويعيش اعمله power مستقل عن الموتور .

لماذا سمى الميكروكنترولر بهذا الاسم ؟؟؟؟

وذلك لأن وظيفته هى التحكم سمى ب controller حيث يتحكم بالدائرة الالكترونية وبما تحوية من عناصر الكترونية وسمى بمايكرو لان حجمة صغير جدا بالنيبة لامكانياته الكبيرة .



هنشتغل ب PIC 16 وأفضل نوع هو PIC 16F877A



هنختار لغة ال C لانها لغة متداولة وسهلة ال MICRO مش يفهم C لذا نستخدم برنامج يحول من ال C CODE الى Machine language ويعرف باسم C compiler .

PIC 16 microcontrollers Datasheet

للتعرف على ال IC اللى هنشتغل بيها نقرأ ال data sheet الخاصة بها نزور موقع www.microchip.com هنختار PIC16F87XA حيث X تعبر عن الارقام الاتية 3 و4 و6 و7 .

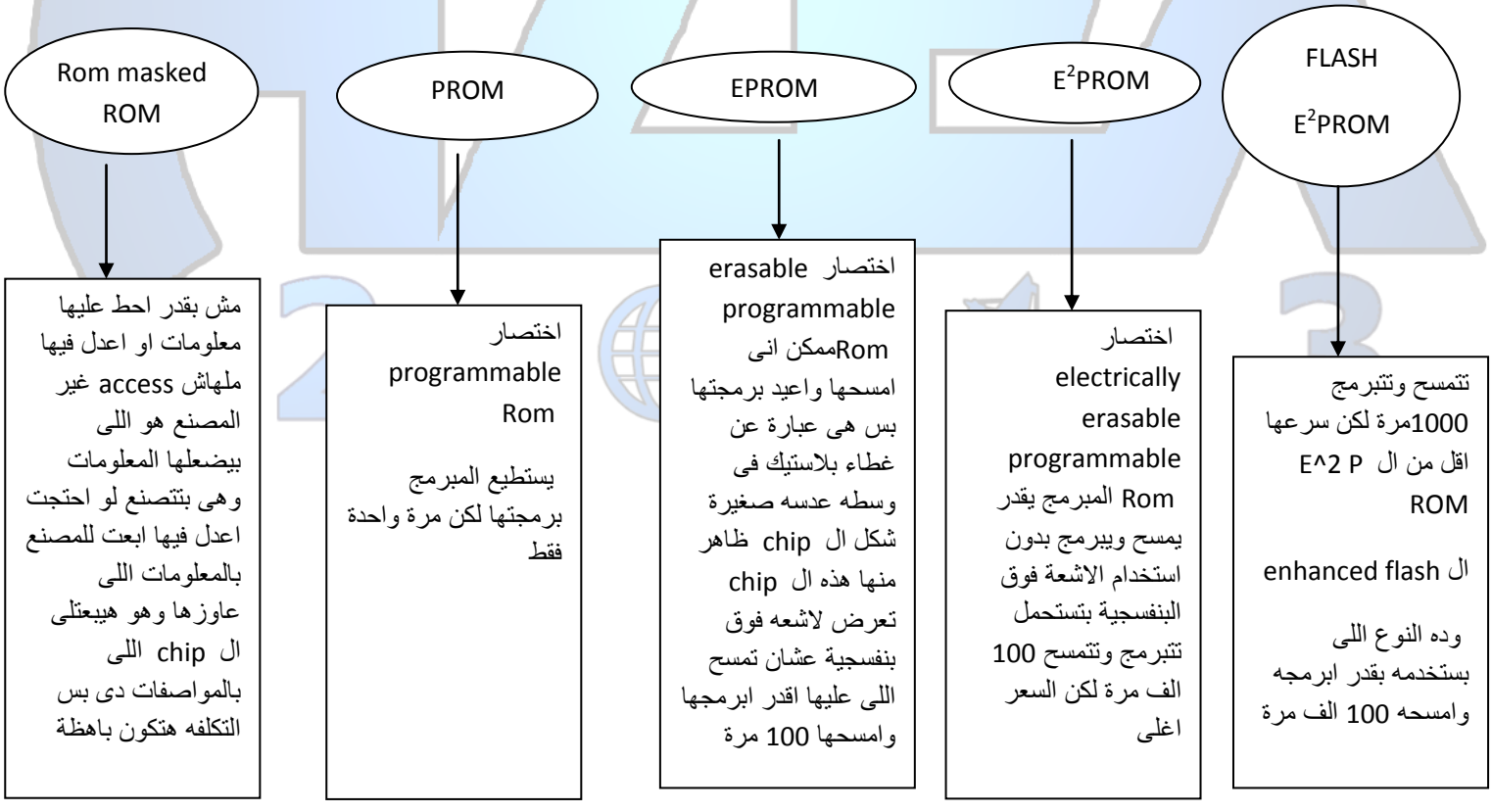
طب ليه وضعنا كذا نوع فى نفس ال Data sheet وذلك لانهم شبه بعض تقريبا فى الخواص .

احنا هنستخدم PIC16F877A وهى عبارة عن 40 pin

هل ينفع أن تكون ال program memory ذاكرة متطايرة ؟؟؟

معنى الذاكرة المتطايرة انه بمجرد فصل ال power فان المعلومات المخزنه تضيع لا الذاكرة المفروض ان تكون nonvolatile غير متطايرة

أنواع الذاكرة فى ال PIC



إمكانيات ال PIC المستخدمة

سنقرأ الداتا شيت لل PIC المستخدمة لنتعرف على بعض خصائصها ومنها :

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
 DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
 Up to 368 x 8 bytes of Data Memory (RAM),
 Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

أولا : ال operating speed لل clock تتراوح من تردد 0 _20 MHZ .

ثانيا : عدد المتغيرات فى ال memory هى 268 وهى كافية لان أغلب تطبيقات الميكرو لا يدخل فيها حسابات .

ثالثا : يوجد منه أنواع 40 أو 44 pin وال pic16f877a واللى هنشتغل بيها عبارة عن 40 pin .

ويوجد بعض ال interfaces الموجودة فى الميكرو والتى توضحها الداتا شيت ومنها :

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
 - Capture is 16-bit, max. resolution is 12.5 ns
 - Compare is 16-bit, max. resolution is 200 ns
 - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
 - Two analog comparators
 - Programmable on-chip voltage reference (V_{REF}) module
 - Programmable input multiplexing from device inputs and internal voltage reference
 - Comparator outputs are externally accessible

يوجد لدى :

1_ 3 timers

2_ serial port for connection with computer

3_ADC with 8 channels

بعض الخصائص الاضافية :

يتم برمجتها ومسحها 100 ألف مرة

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical

الخصائص الكهربية للميكرو :

17.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias	-55 to +125°C
Storage temperature	-65°C to +150°C
Voltage on any pin with respect to Vss (except VDD, MCLR, and RA4)	-0.3V to (VDD + 0.3V)
Voltage on VDD with respect to Vss	-0.3 to +7.5V
Voltage on MCLR with respect to Vss **(Note 2)**	0 to +14V
Voltage on RA4 with respect to Vss	0 to +8.5V
Total power dissipation **(Note 1)**	1.0W
Maximum current out of Vss pin	300 mA
Maximum current into VDD pin	250 mA
Input clamp current, IIK (VI < 0 or VI > VDD)	± 20 mA
Output clamp current, IOK (VO < 0 or VO > VDD)	± 20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sunk by PORTA, PORTB and PORTE (combined) **(Note 3)**	200 mA
Maximum current sourced by PORTA, PORTB and PORTE (combined) **(Note 3)**	200 mA
Maximum current sunk by PORTC and PORTD (combined) **(Note 3)**	200 mA
Maximum current sourced by PORTC and PORTD (combined) **(Note 3)**	200 mA

Note 1: Power dissipation is calculated as follows: Pdis = VDD x {IDD - ∑ IOH} + ∑ {(VDD - VOH) x IOH} + ∑(VOl x IOL)

2: Voltage spikes below Vss at the MCLR pin, inducing currents greater than 80 mA, may cause latch-up.

هى الخصائص الكهربية التى بتعامل بها مع الميكرو بمعنى الجهد اللى بدخلة للميكرو وكذلك التيار الداخل والخارج من الميكرو .

VDD (positive voltage)>>>>>>>>>>>>-0.3_7.5

الجهد الداخل لابد أن يكون فى هذا ال range والا سيحرق الميكرو .

VSS (negative voltage)>>>>>>>>>>>>>zero

التيار سواء sunk يعنى داخل أو sourced يعنى خارج لا يزيد عن 25mA .

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

مش بحتاج متغيرات كتير لان معظم تطبيقات الميكرو مش بتحتاج RAM كبيرة

I/O عدد ال

من النوع 2TIMER عندى 8BIT
من النوع 1 TIMER وعندى 16 BIT

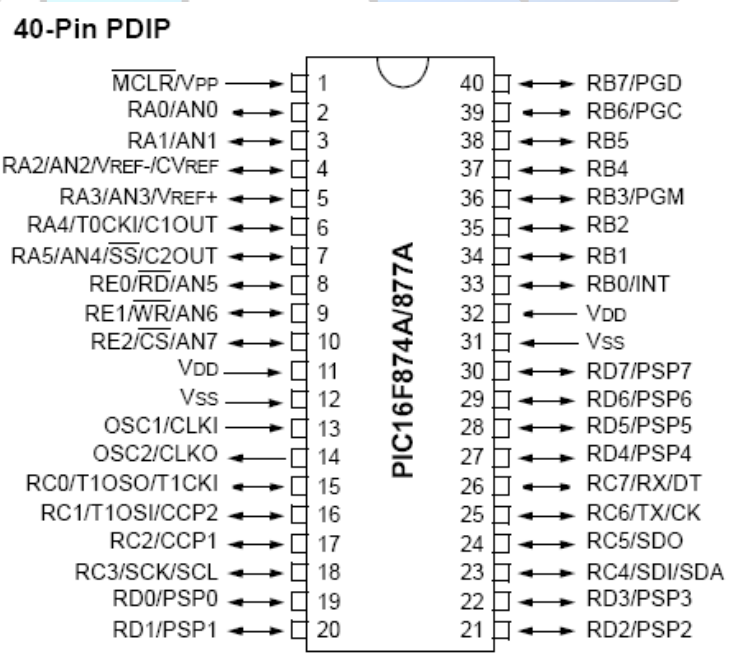
يوجد اشكال من هذا ال PIC

PDIP

الأرجل تكون فى صفين 20 يمين و20 يسار

Through hole mount

بيكون فيه hole بتدخل رجل ال IC منها وتتلحم من الناحية الاخرى



TQFP

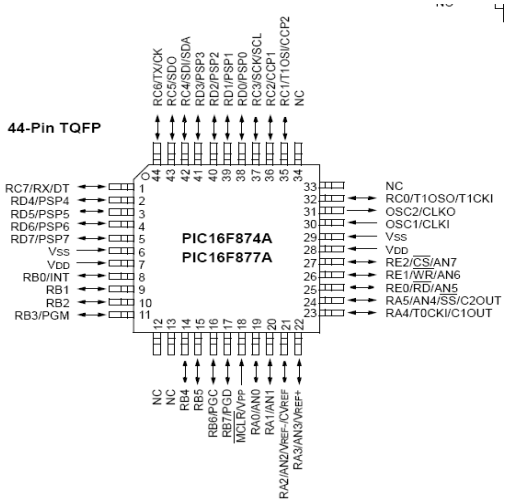
Thin quad flat package

تكون مثبته على السطح

Surface mount

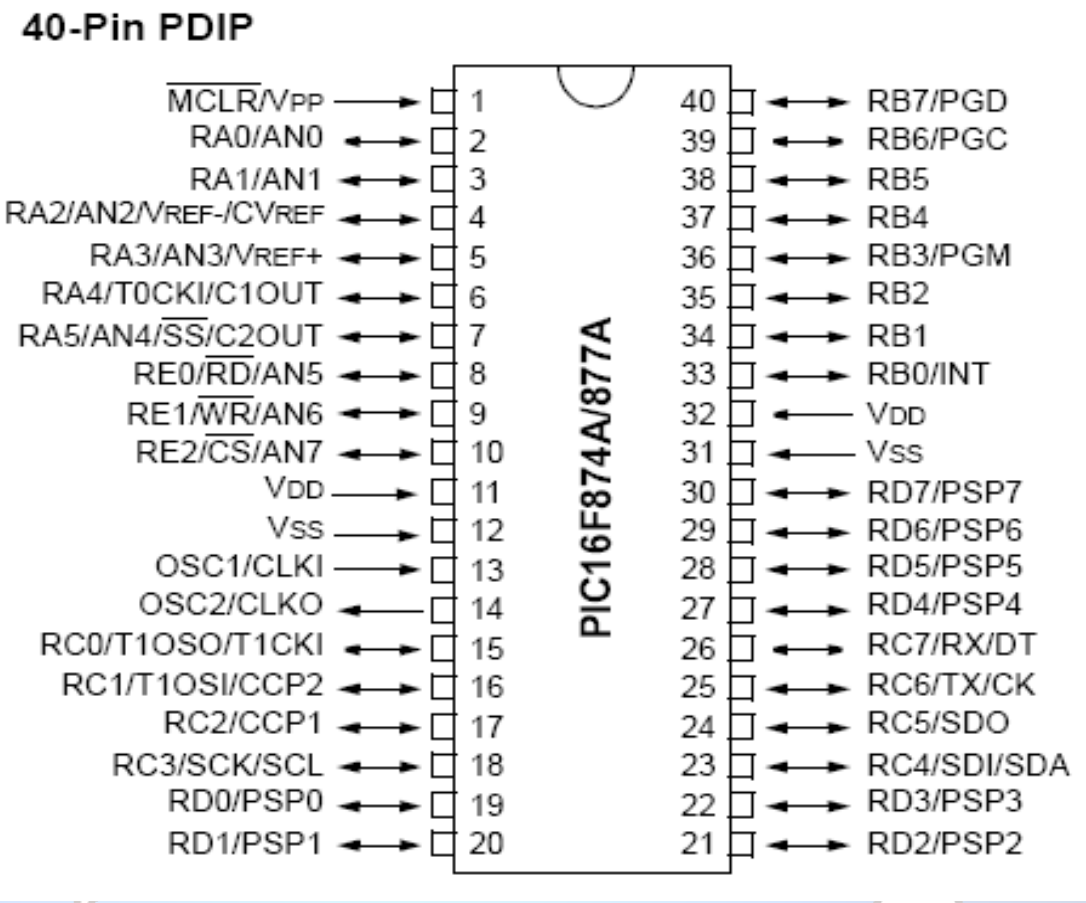
مش بتنفع فى الشغل العادى لانها محتاجه PCB

يوجد فى هذا النوع 4 رجول مش مستخدمه مكتوب عليها NC لان ال standard لهذة ال package فى



التصنيع انها 44 pin .

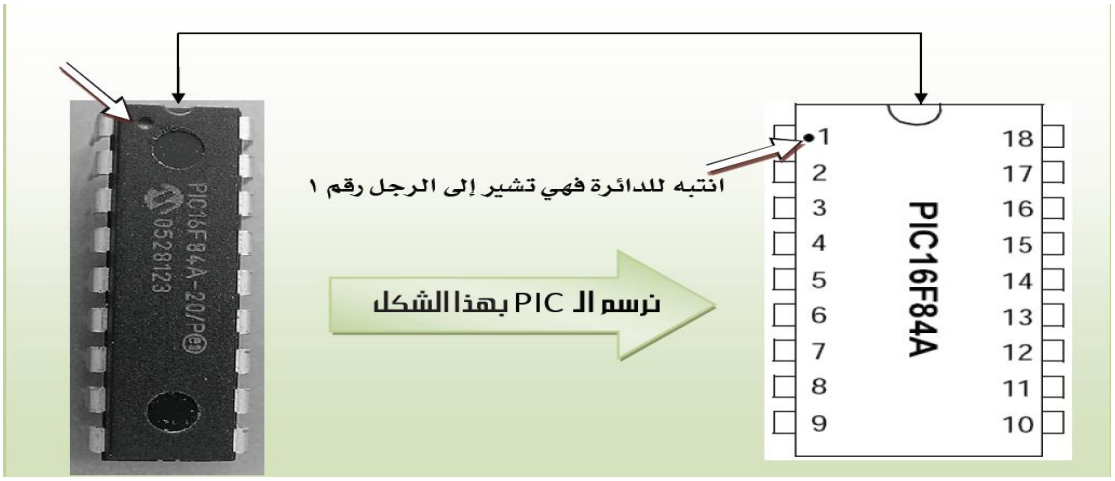
التعرف على ال PIC



Specifications of microcontroller "data sheet"

1_How to read micro pins.

1_ نصف الدائرة اللى فى اول ال IC عبارة عن notch اى ان هذا الجنب هو رأس ال IC pin رقم 1 هى االتى على يسار ال notch ويوجد أحيانا hole بجانب pin1 .



2_Oscillator 20MHZ

2_ يوجد فى التركيب الداخلى للميكرو counters & registers ومن المعروف انها تعمل ب clock وهذة ال clock عبارة عن digital signal لها تردد معين أقصى قيمة تردد بداخل الميكرو pic16f877a هو 20MHZ ولهذا عند شراء كريستاله لكى تولد clock pulses نشتريها 20MHZ أو أقل والشائع استخدامه هو ال 4MHZ & 8MHZ .

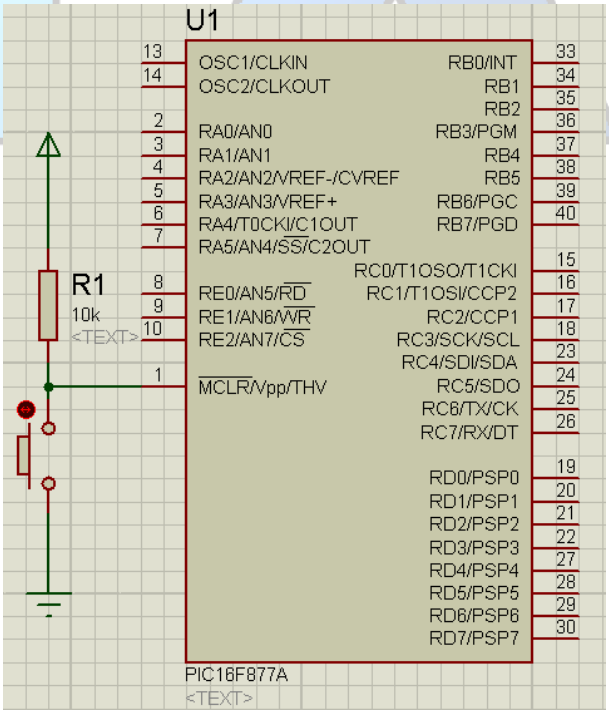


3_Currents 25mA

3_أقصى تيار داخل يتحمله الميكرو وأقصى تيار خارج منه هو 25mA وهذا مهم جدا معرفته وخاصة فى دوائر ال relay interface وسيتم ذكرها لاحقا .

4_ Pins description.

Pin1 :



مكتوب عليها MCLR فهى تشير إلى MASTER CLEAR بمعنى انه بيعمل reset يعنى لو ال PIC بيقوم بتنفيذ برنامج معين وعملت reset هيقوم بتنفيذ البرنامج من الاول وهذة ال pin active low بمعنى أنها لكى تعمل اضعها على low voltage اى (0v) لكن لكى لا تعمل اضع عليها VCC يعنى 5v طب دلوقت انا عاوز ال pin1 لما احب اعمل reset توصل بالارضى وبعد كده تكون موصله بال VCC علطول لذا يتم توصيلها بهذا الشكل

لما اضغط على ال push button يعمل reset لان ال pin1 هيكون واصل ارضى لما اشيل ايدى من على ال button هتكون ال pin float بمعنى اى noise ممكن تخليها high او low بس انا عاوز لما اشيل ايدى تكون موصله بال VCC لذا اضع مقاومة مع ال VCC التيار المار فى المقاومة قليل جداً لان مقاومة الدخل لل pin كبيرة جداً يعنى ال drop اللى على المقاومة تقريبا zero يعنى الفولت على pin 1 هيكون VCCوتسمى المقاومة فى هذة الحالة pull up resistance يعنى جعلتها بدل ما هى float جعلتها high .

ال push button دا اختيارى ممكن احطه او لا واستعمل لما الجهاز يهنج زر ال power اطفى منه الجهاز وافتحه تانى كده انا عملت reset للجهاز اما المقاومة اجبارى ان اضعها عشان تكون pin 1 موصله بال VCC علطول .

33 I/O PINS

1_PORT A → 6 PINS from RA0:RA5

2_PORT B → 8 PINS from RB0:RB7

3_PORT C → 8 PINS from RC0:RC7

4_PORT D → 8 PINS from RD0:RD7

5_PORT E → 3 PINS from RE0:RE2

- يمكن معامله ال PIN على انهم Individual كل واحده لوحدها او على انهم مجموعات المجموعه A مثلا تخرجلى خرج كذا
- ال PIN ممكن تكون Analog (ANO) اوdigital (RAO) مش ينفع تكون الاتنين مع بعض.
- توصيل ال PORTS يختلف حسب برمجة الدائرة لكن ال 7 PINS وهما (1/32/11/12/31/14/13) لهم توصيلة ثابته .
- يفضل استخدام PORTB اثناء التوصيل لان أرجلة ورا بعضها .
- PORTA & PORTE يستخدموا لقراءة قيم analog & digital

أما (PORTD,PORTC,PORTB) يستخدموا لقراءة قيم digital فقط .

- ال programmer هو اللى بيحدد ال pin هتستخدم ك in أو out أو هتستخدم ك analog أو digital .

Pin26 مكتوب عليها RC7/RX وكذلك PIN25 مكتوب عليها RC6/TX طب ايه هما ال RX وال TX هما يدلان على وظيفة اخرى تقوم بها ال PIN وهى توصيل الميكرو بالكومبيوتر وذلك عن طريق استخدام ال SERIAL PORT من خلال طرفى ال SERIAL PORT وهما ال transmit وال receive وأوصلهم بال TX/RX وبكده فأنا ممكن استخدم ال PIN 26 و PIN 25 ك digital in أو digital out أو إنى اربط الكومبيوتر بيهم .

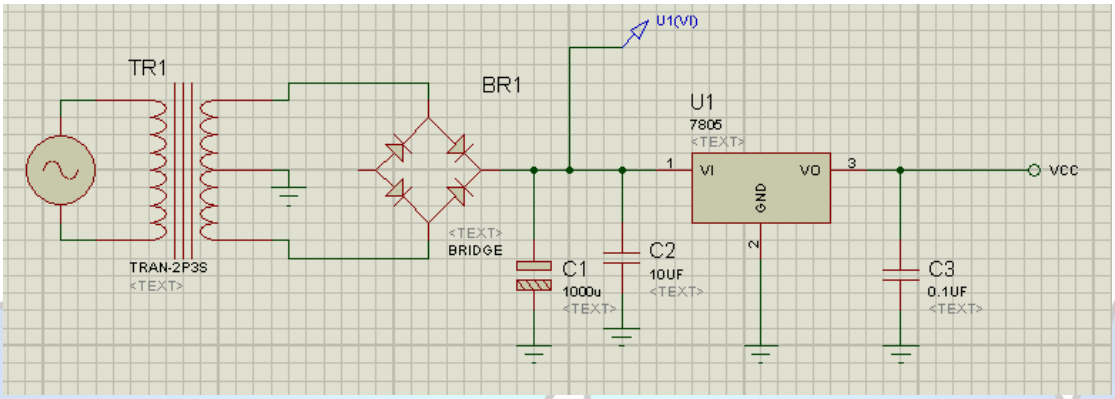
PIN32/PIN 11

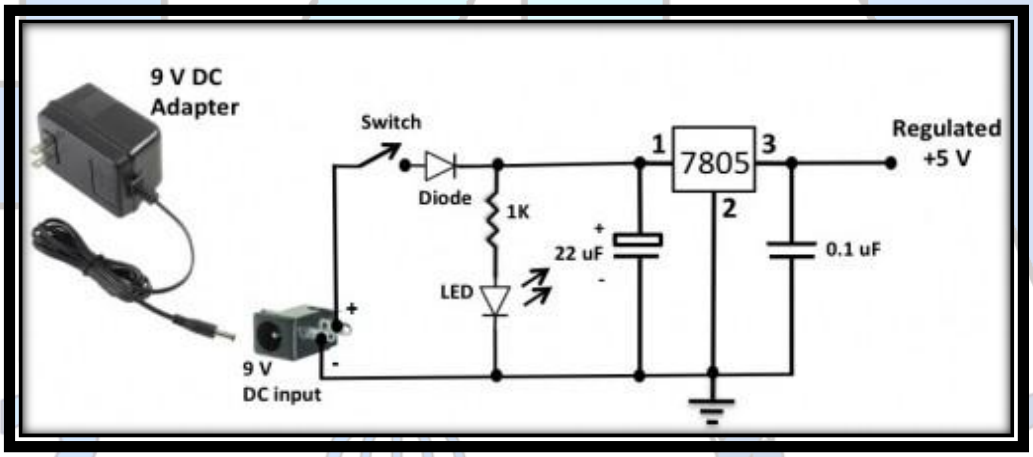
توصل بال VDD اى POWER SUPPLY+ وبيكون 5V

PIN12 /PIN31

توصل بال VSS يعنى الارضى

- المفروض اقل فولت VDD يدخل من 0.3:7.5- يعنى لو عكست البولارتى لل POWER ودخلت السالب على ال VDD فانه الميكرو يبوظ
- المفروض ال SUPPLY مش يقل عن 4 ولا يزيد عن 5.5 طب لو وضعت 3V على VDD فان ال IC مش يبوظ لكن مش يشتغل.
- المفروض اضع بين الرجل 11 و12 POWER SUPPLY ثابت بهذا الشكل او عن طريق adaptor





يوصل بين الرجل 11 و12 مكثف سيراميكى 0.1uf يشيل ال noise اللى جايه من اسلاك التوصيل لل supply وكذلك بين 31 و32 ..

ممكن بدل ال adaptor بطارية 9v .

ممكن شاحن موبيل يقطع طرفة ويوصل طرفية بروزته لسهولة تركيبة فى الدائرة .

ممكن power supply of computer وهو الافضل لانة يعطى تيار عالى لأن فيه component فى الدائرة بتسحب تيار عالى زى الريلاى والمواتير فلازم اتاكد ان التيار اللى بيديه الباور اللى شغال بيه كافى لتشغيل الكومبونانت اللى موجودة فى الدايرة ويتم حساب ذلك من العلاقة

P=V* I

بقسم جهد تشغيل الريلاى على الباور بتاعته احصل على التيار اللى بيشغله.

Pin 13 /pin14

OSC1&OSC2

طب مش المفروض ان ال oscillator built in بداخل ال pic لأ فى جوه ال PIC وخارجها .

انواع ال OSCILLATOR

١- RC Oscillator وهو عبارة عن مقاومة ومكثف (R و C) وهذا النوع يتميز بوفرته فيمكنك الحصول عليه بسهولة كما يتميز برخص ثمنه ، ويستخدم في التطبيقات والمشاريع التي لا نحتاج إلى دقة الوقت فيها . لهذا عند استخدامه لن نتعامل مع الميكروثانية .

٢- Crystal oscillator (كريستال) هذا النوع هو عبارة عن قطعة الكترونية تنتج تردد معين وهذا النوع يستخدم في المشاريع التي نحتاج فيها إلى دقة عالية في الوقت فهو مناسب لهذا الغرض . وغالبا ما نجد التردد الخاص بالكريستال مكتوب على جسم القعطة .





بعض أشكال الكريستال

إذا كنت تستخدم كريستال فغالبا ما ستجد تردده مكتوب عليه . أما إذا كنت تستخدم RC OSC فإن تردده يعتمد على عدة عوامل أهمها قيمة المقاومة (R) وسعة المكثف (C) .

ويمكنك حساب التردد في تلك الحالة من المعادلة التالية :

$$\text{التردد} = \frac{1}{4.2 \times R \times C}$$

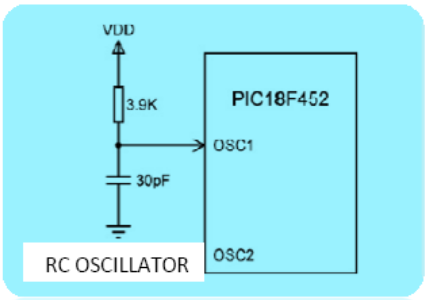
والتردد المحسوب هو تردد تقريبي لأن هناك عوامل أخرى يعتمد عليها التردد مثل درجة الحرارة .

وإليك الآن جدول ببعض القيم المستخدمة لهذا النوع من المذبذبات حيث يوضح الجدول قيمة المقاومة وسعة المكثف والتردد الناتج (التردد التقريبي) :

قيمة المقاومة	سعة المكثف	تردد المذبذب
3.3 K	22 pF	3.3 MHz
4.7 K	22 pF	2.3 MHz
10 K	22 pF	1.08 MHz
3.3 K	30 pF	2.4 MHz
4.7 K	30 pF	1.7 MHz
10 K	30 pF	0.793 MHz

لهذا عندما كنا نستخدم مكثف ٢٢ بيكو فاراد ومقاومة ١٠ كيلو أوم فإننا كنا نكتب التردد 1.08MHz .

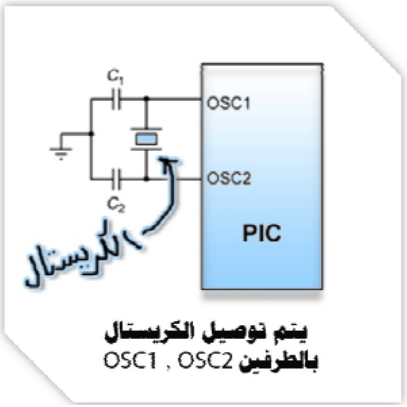
للتذكير : كنا نوصل المذبذب RC بهذه الطريقة أيًا كان نوع البك المستخدم المهم أن نقوم بالتوصيل بالطرف المكتوب بجواره OSC1 كما بالشكل الموضح :



بالطبع يمكنك تغيير قيمة المقاومة والمكثف لتغيير التردد .

كيف نقوم بتوصيل الكريستال ؟

عند توصيل الكريستال فإننا نوصله بالطرفين OSC1 ، OSC2 ومن المهم أن نقوم بتوصيل مكثفين مع الكريستال ، كما يوضح ذلك الشكل التالي :



ويتم التوصيل بالأرضي كما هو موضح ولمعرفة الطرفين OSC1,OSC2 قم بفتح الداتاشيت Datasheet واذهب إلى pin diagram ، فستجد صورة توضح لك أرقام كل الأطراف واسم كل طرف ومنه ستجد OSC1,OSC2

وتوصل الكريستاله بهذين الطرفين

السؤال المهم الآن هو ما هي قيم تلك المكثفات المستخدمة مع الكريستال؟

إن قيم المكثفات تعتمد على شيئين:

١- تردد الكريستال المستخدم . ٢- الـ CRYSTAL MODE والمقصود به هل هو XT أم HS أم ..

والجدول التالي يوضح القيم التي يمكن اختيارها للمكثفات .

MODE	تردد الكريستال	قيمة كل مكثف
XT	100 KHz	100-150 pF
XT	200 KHz	22-68 pF
XT	2 MHz	15-33 pF
XT	4 MHz	15-33 pF
HS	4 MHz	15-33 pF
HS	8 MHz	15-33 pF
HS	10 MHz	15-33 pF
HS	20 MHz	15-33 pF

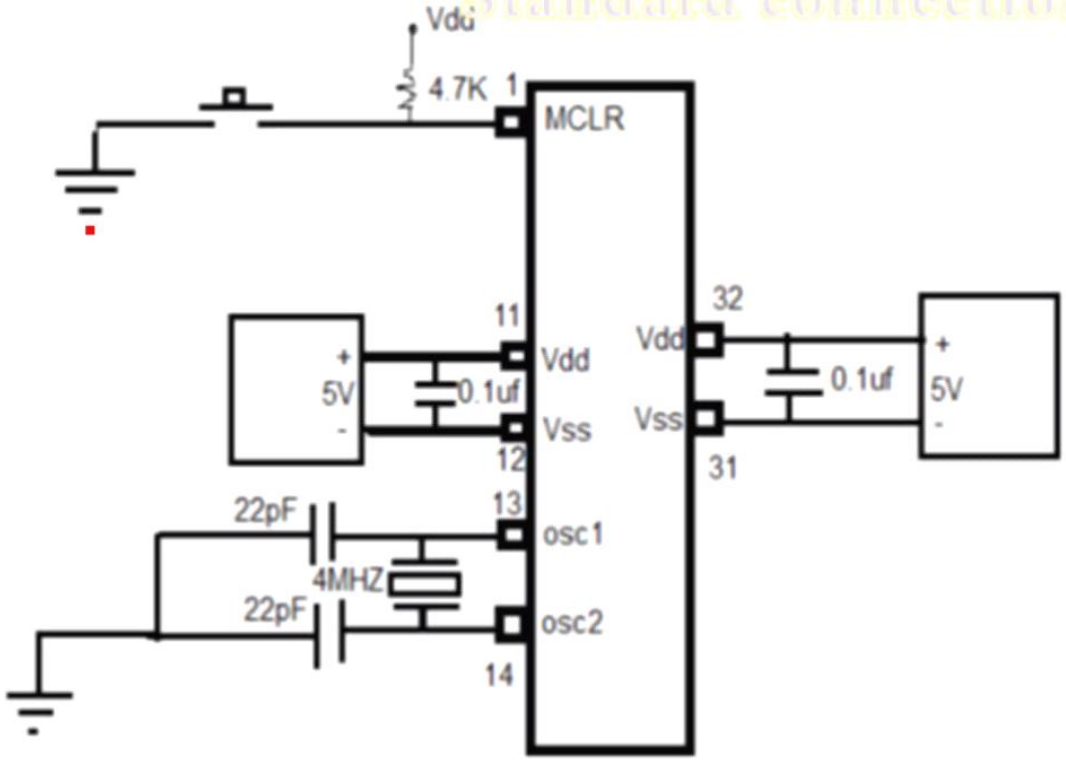
ستلاحظ من الجدول أن الكريستال ٨ ميجا هيرتز مثلا يمكنك أن تختار له مكثفين قيمة كل واحد منهما قد تكون ١٥ بيكو فاراد (وهي قيمة محصورة بين 15-33Pf) أو تجعل قيمة كل منهم تساوي ٢٢ بيكوفاراد لأن ٢٢ قيمة محصورة بين 15-33pF . إذن أنت لديك العديد من الاختيارات وعمومًا كلما زادت سعة المكثف زاد استقرار المذبذب ولكن في نفس الوقت يزداد الوقت اللازم لكي يبدأ

متى يفضل استخدام ال RC circuit ؟؟؟؟؟؟

لما تكون حسابات الوقت مش شرط تكون دقيقة زى ال open door وذلك لانه رخيص اما لو فرق معايا الزمن زى ال stop watch استخدم crystal وبروح اشترى crystal بقيمة التردد اللى عوزة .

الشكل النهائى للتوصيل

ودائرة ال POWER هى دائرة ال adaptor المذكورة سابقا



Standard connection

Chapter (2)

Mikro C



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

C Compiler

Software requirements:

1_ PROTEUS

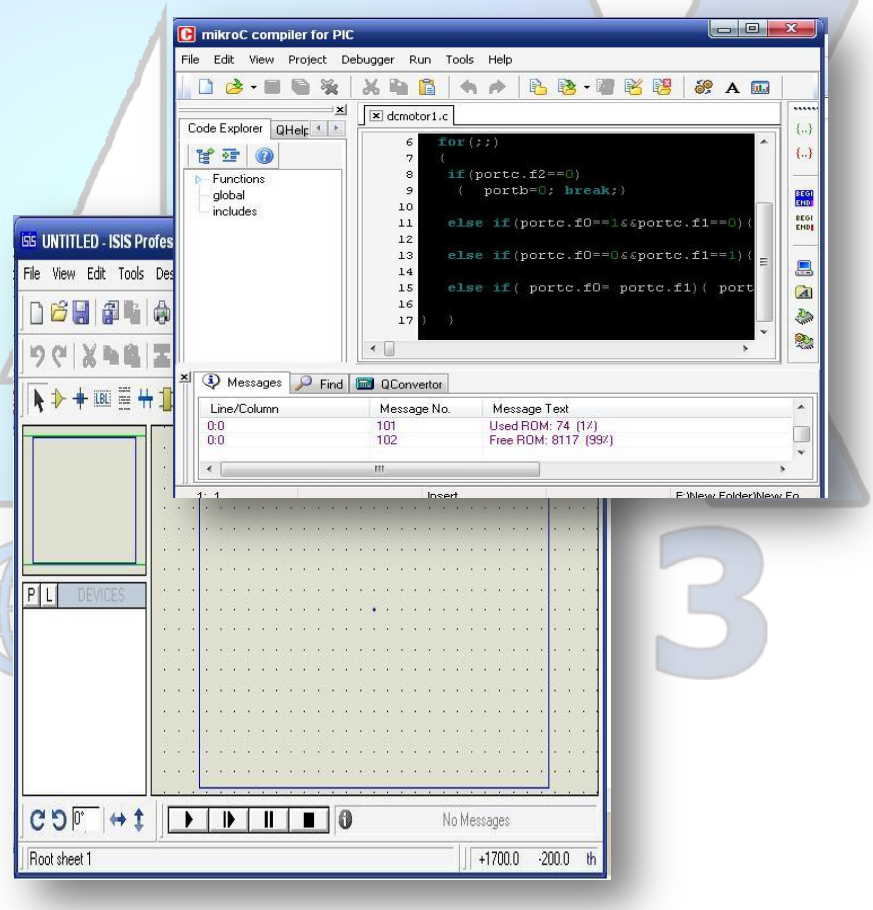
SIMULATION يستخدم لعمل

2_ MICROC

. PIC ع ال بحرقة اللى CODE برنامج بستخدمة لكتابة ال

3_burner program

برنامج باستخدمة عشان أضع البرنامج اللى كتبته على الميكروكنترولر



HARDWARE REQUIREMENTS:

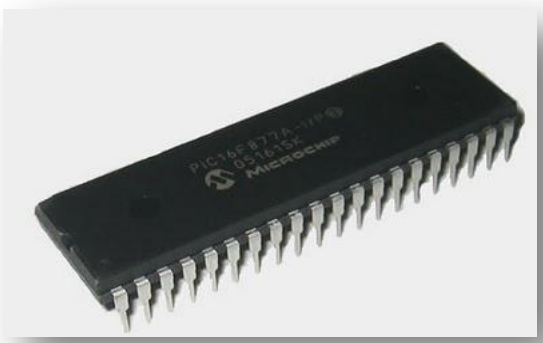
1_COMPUTER

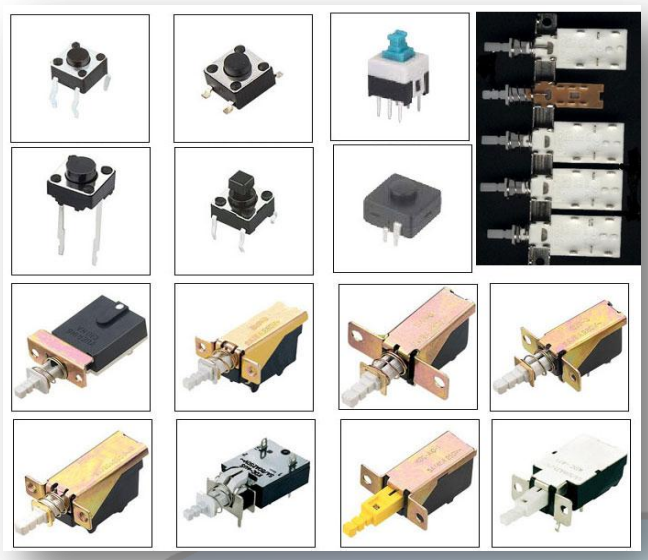
2_PROGRAMMER

3_COMPONENTS

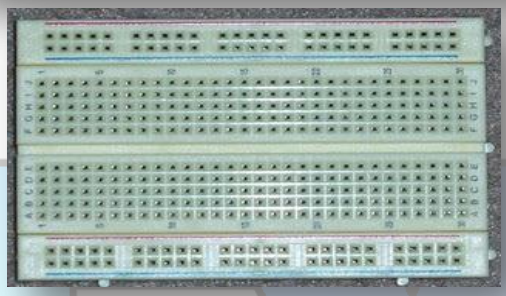
Such as (pic16f877a/crystal 4MHZ/ LCD 2 × 16/LEDS/PUSH BUTTONS/RESISTORS (330/10K/1KΩ)/DC MOTOR 12V/RELAY 12V/transistor bc547/ Pin header /sevensegment)

4_ POWER SUPPLY









HOW TO CHOOSE MICROCONTROLLER:

بختار نوع الميكرو اللى عاوز استخدمة فى تطبيق معين بناء على مجموعة من ال parameters :

1_NUMBER OF I/O PIN

احدد عدد ال inputs عدد ال SWITCHES وعدد ال sensors وأيضا عدد ال outputs عدد ال LEDS وال relay وال seven segment وهكذا.

2_ ANALOG INPUT

لو هحتاج ادخل كذا اشارة analog هختار ميكرو له analog channels كتير .

3_MEMORY SIZE

ال FLASH memory اللى بكتب فيها البرنامج باقدر اعرف أنا عاوز قد ايه منها بعد كتابة البرنامج على ال c compiler وعمل compile للبرنامج اللى كتابته على برنامج ال micro c بيحدد أنا عاوزقد ايه memory واختار على اساسها الميكرو لابد ان تكون ال memory size للميكرو تكفى ال size للبرنامج اللى عملته .

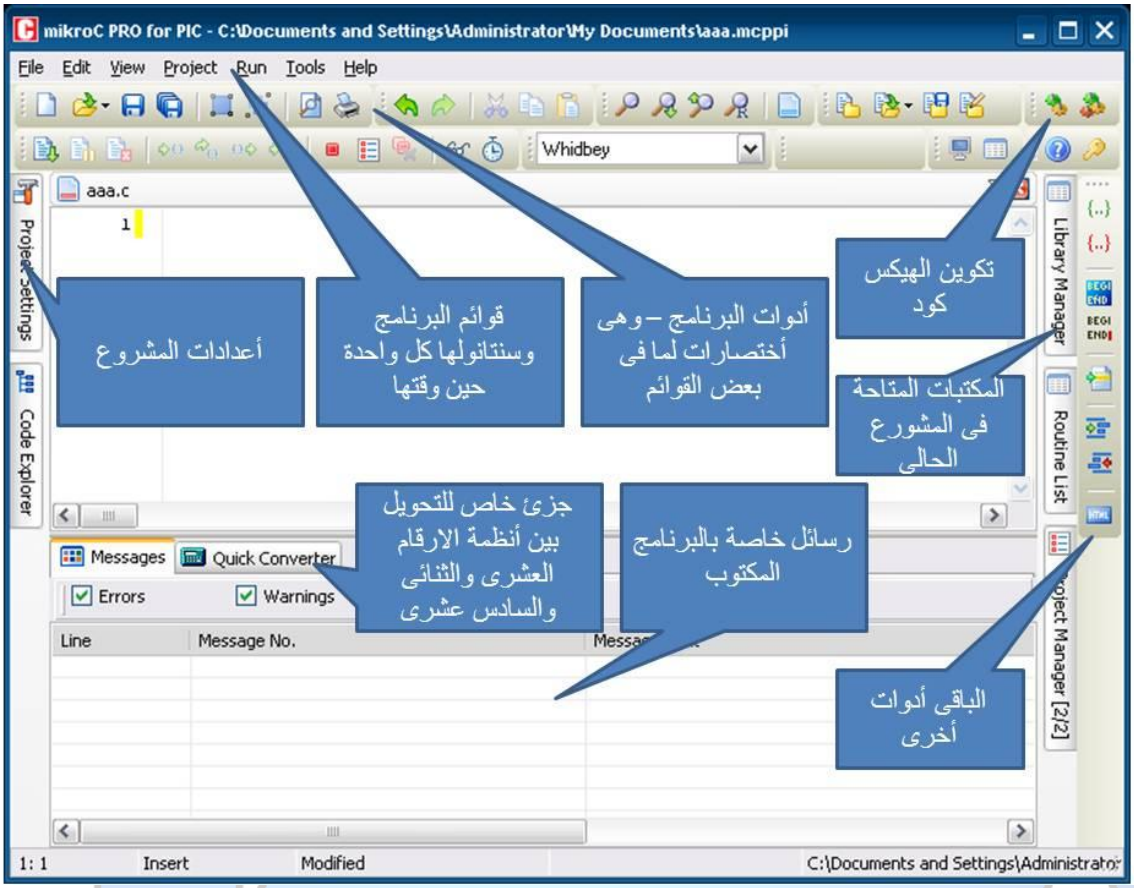
4_ Internal oscillator

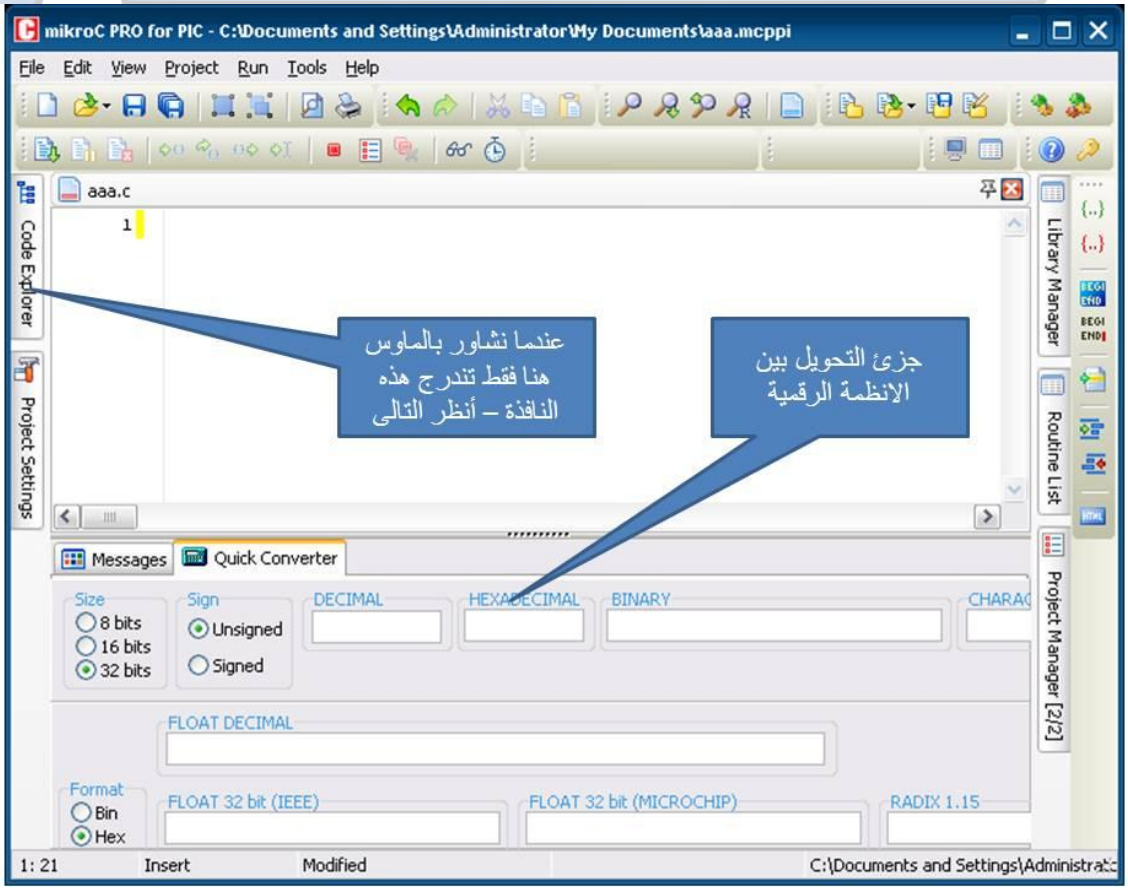
5_ Interfaces
such as (ADC / serial port /timer…….)

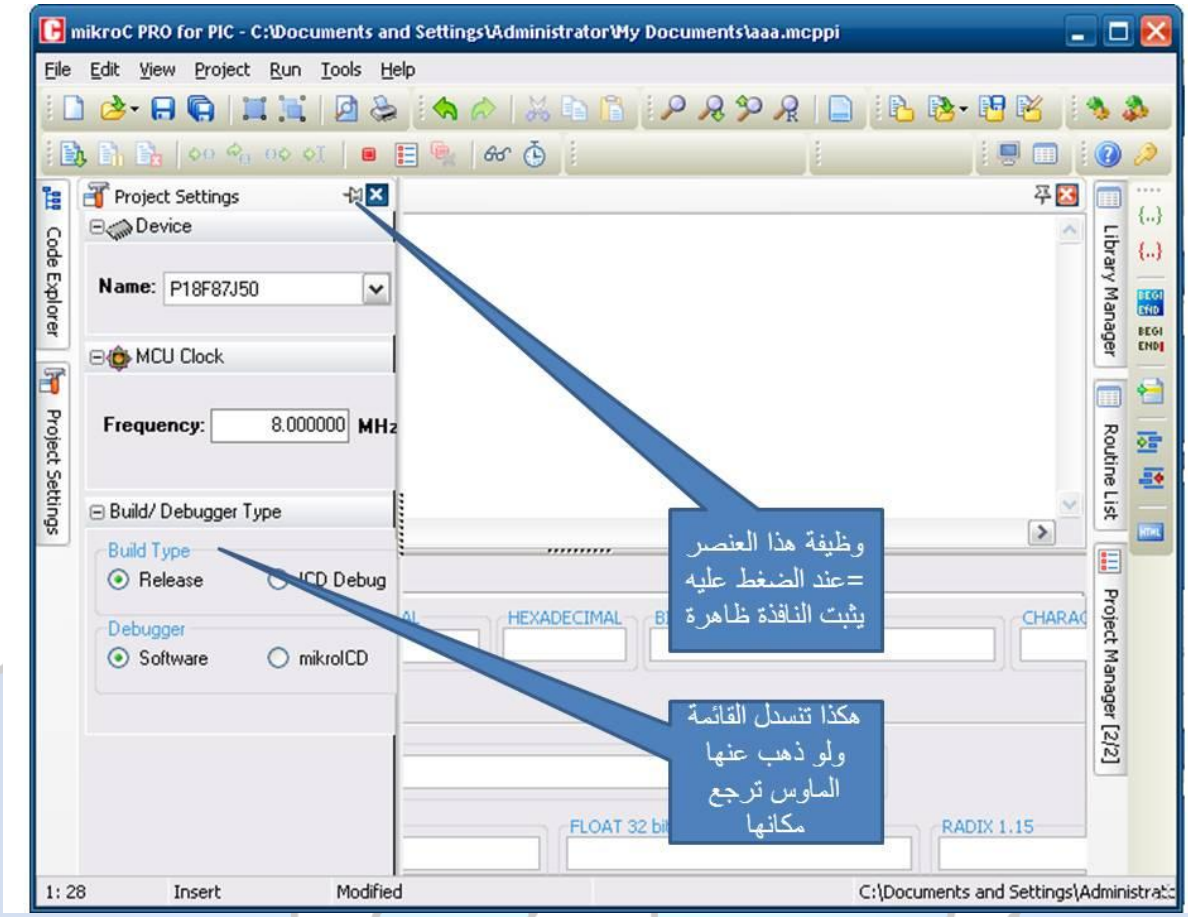
6_ Price

بعد تسطيب البرنامج.....

واجهة البرنامج :

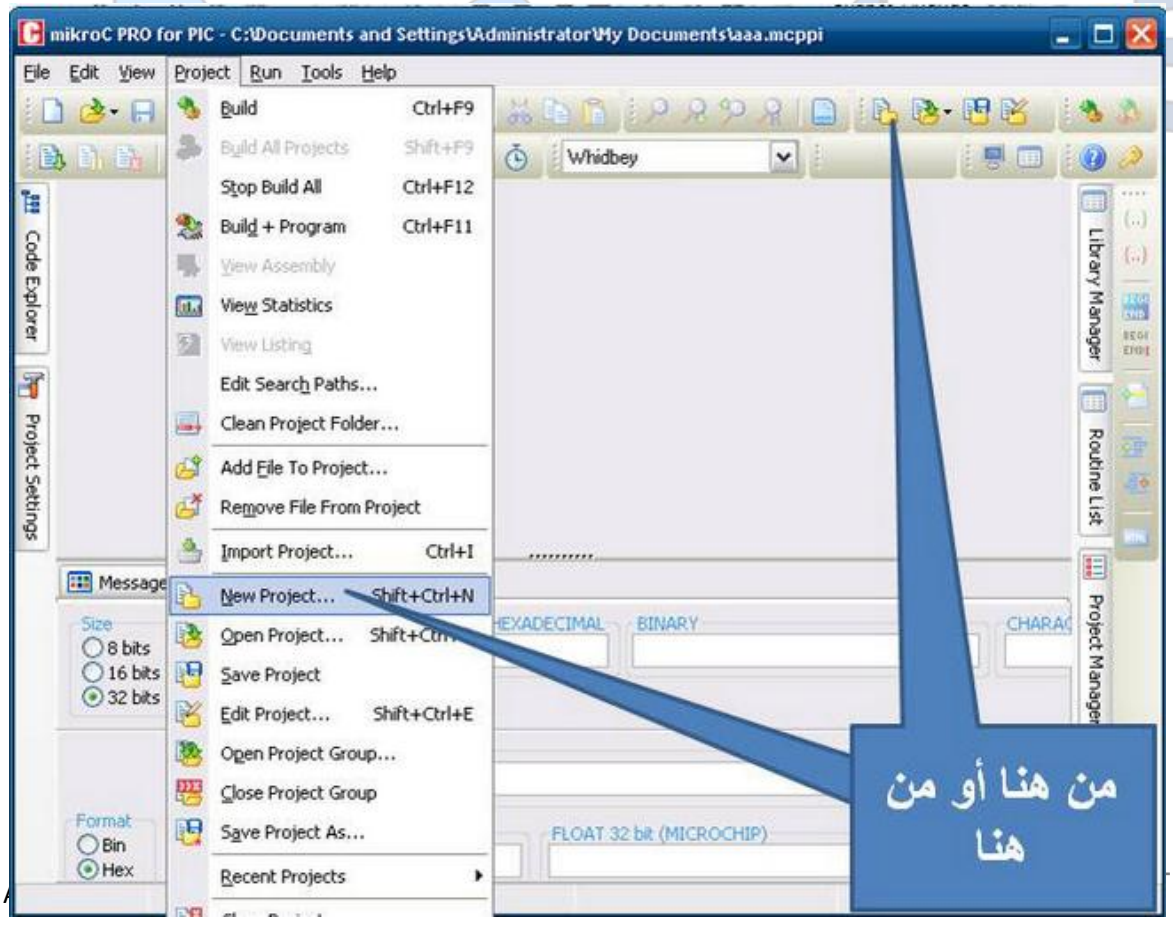




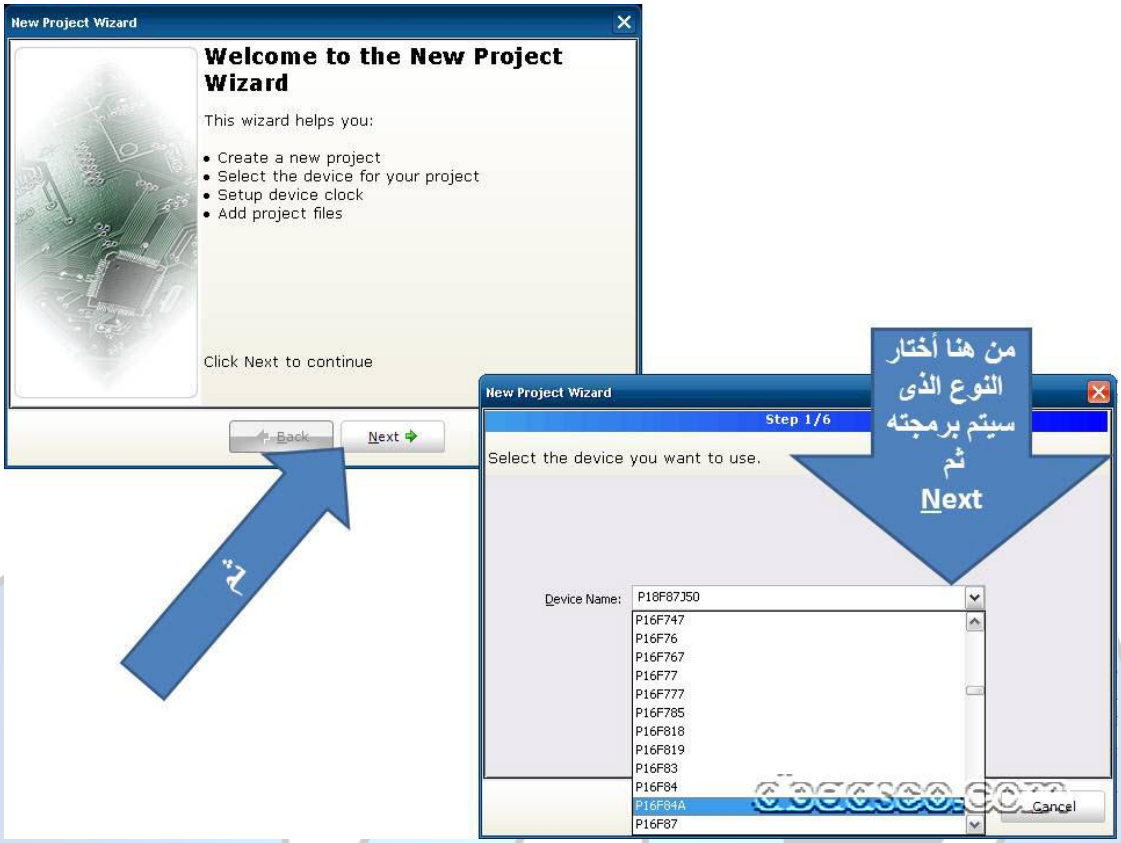


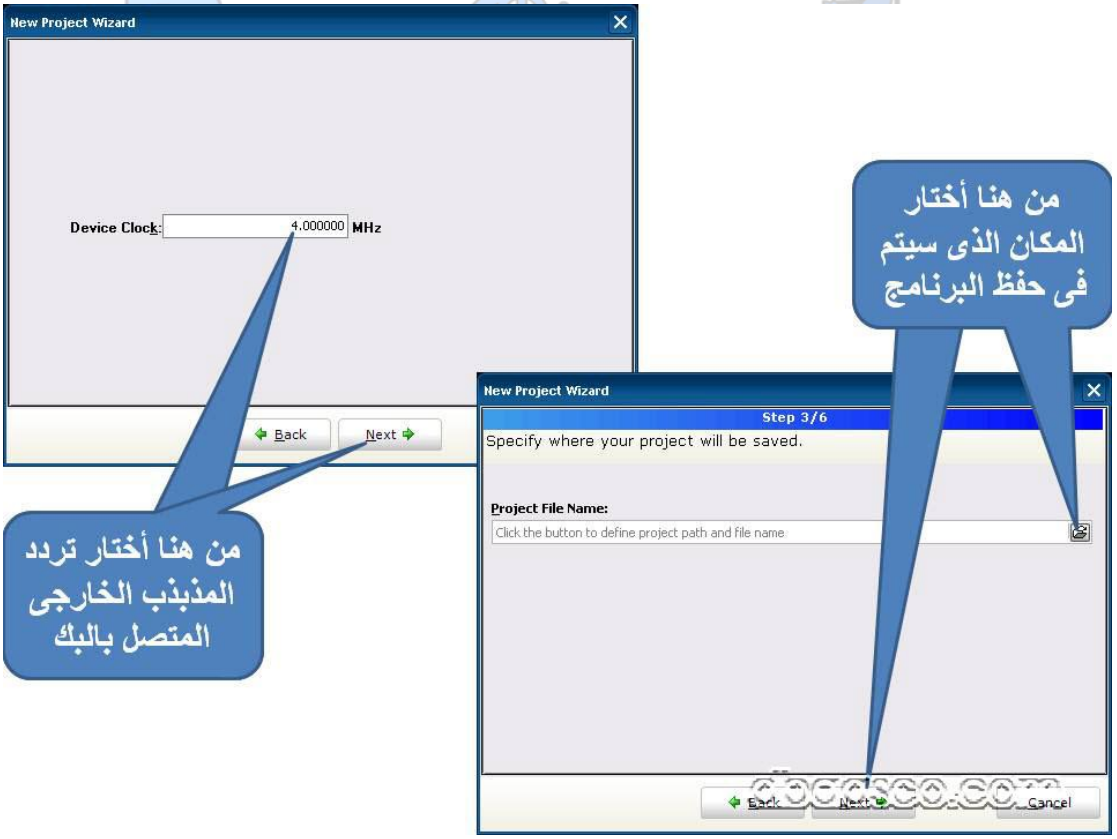
إنشاء مشروع جديد :

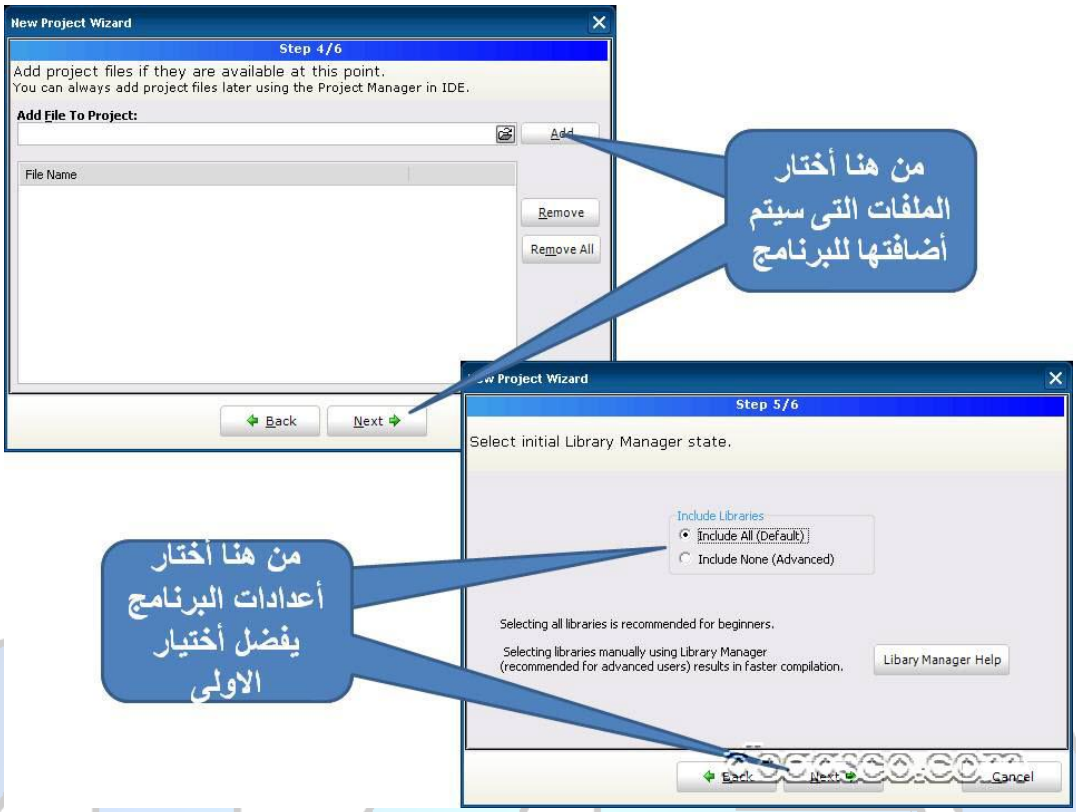
1_من قائمة PROJECT اختار NEW PROJECT

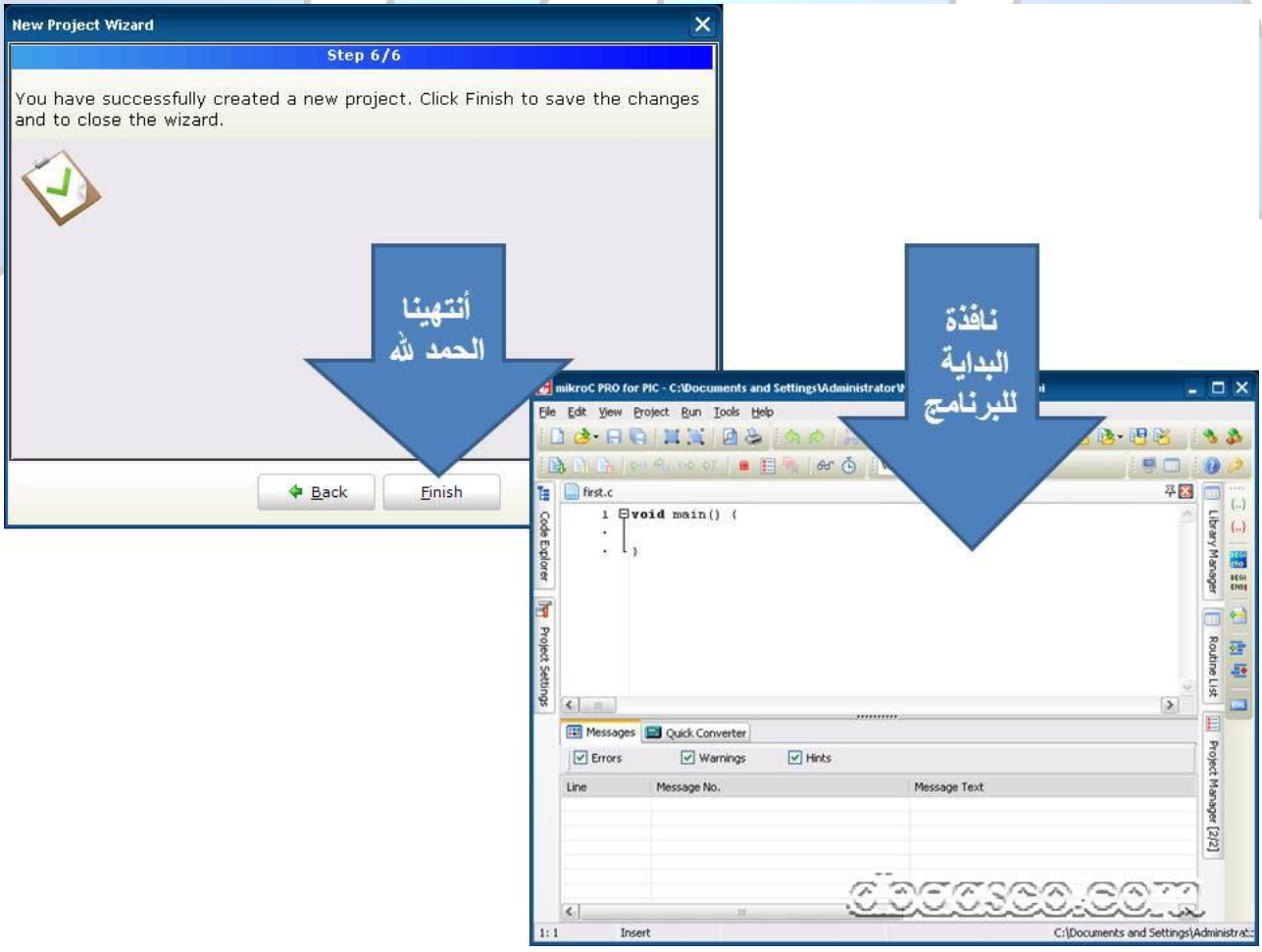


2_ سيظهر لك نافذة اضغط NEXT ثم تظهر النافذة الاتية

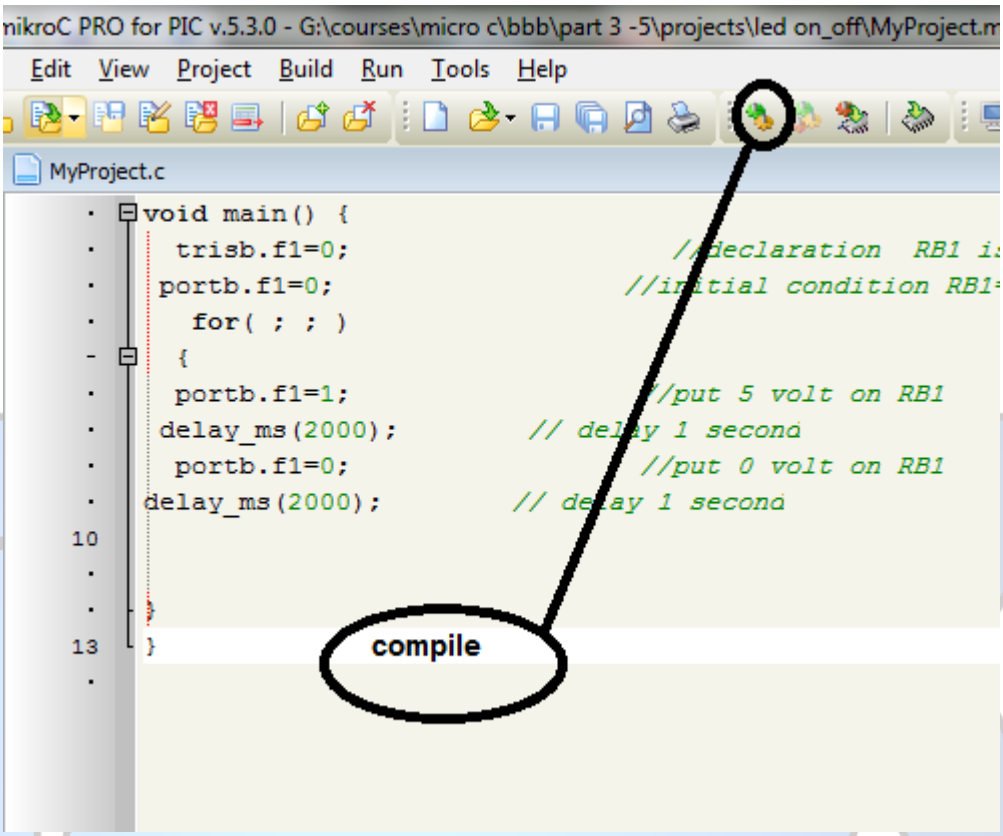





New Project Wizard
Step 4/6
Add project files if they are available at this point.
You can always add project files later using the Project Manager in IDE.
Add File To Project:
Add
File Name
Remove
Remove All
Back
Next
من هنا أختار الملفات التى سيتم أضافتها للبرنامج
Step 5/6
Select initial Library Manager state.
Include Libraries
Include All (Default)
Include None (Advanced)
Selecting all libraries is recommended for beginners.
Selecting libraries manually using Library Manager (recommended for advanced users) results in faster compilation.
Libary Manager Help
Back
Next
Cancel
من هنا أختار أعدادات البرنامج يفضل أختيار الاولى

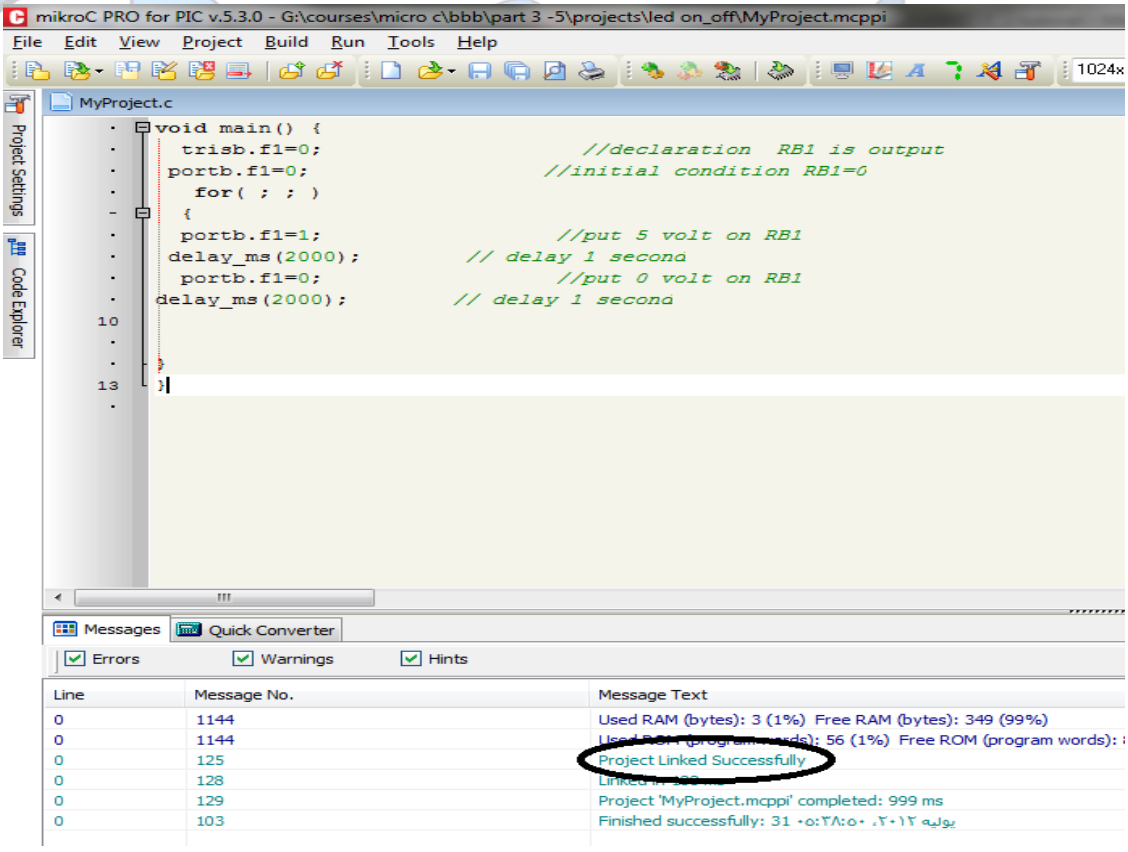

New Project Wizard
Step 6/6
You have successfully created a new project. Click Finish to save the changes and to close the wizard.
Back
Finish
أنتهينا الحمد لله
نافذة البداية للبرنامج
mikroC PRO for PIC - C:\Documents and Settings\Administrator
File Edit View Project Run Tools Help
first.c
void main() {
Code Explorer
Project Settings
Library Manager
Routine List
Project Manager [2/2]
Messages
Quick Converter
Errors
Warnings
Hints
Line
Message No.
Message Text
Insert

قم بكتابة البرنامج فيها ثم اضغط build أو compile للتأكد من صحة البرنامج ولانها مسئوله عن عمل GENERATE للملف ال hex والذى يتم تحميلة على الميكرو .

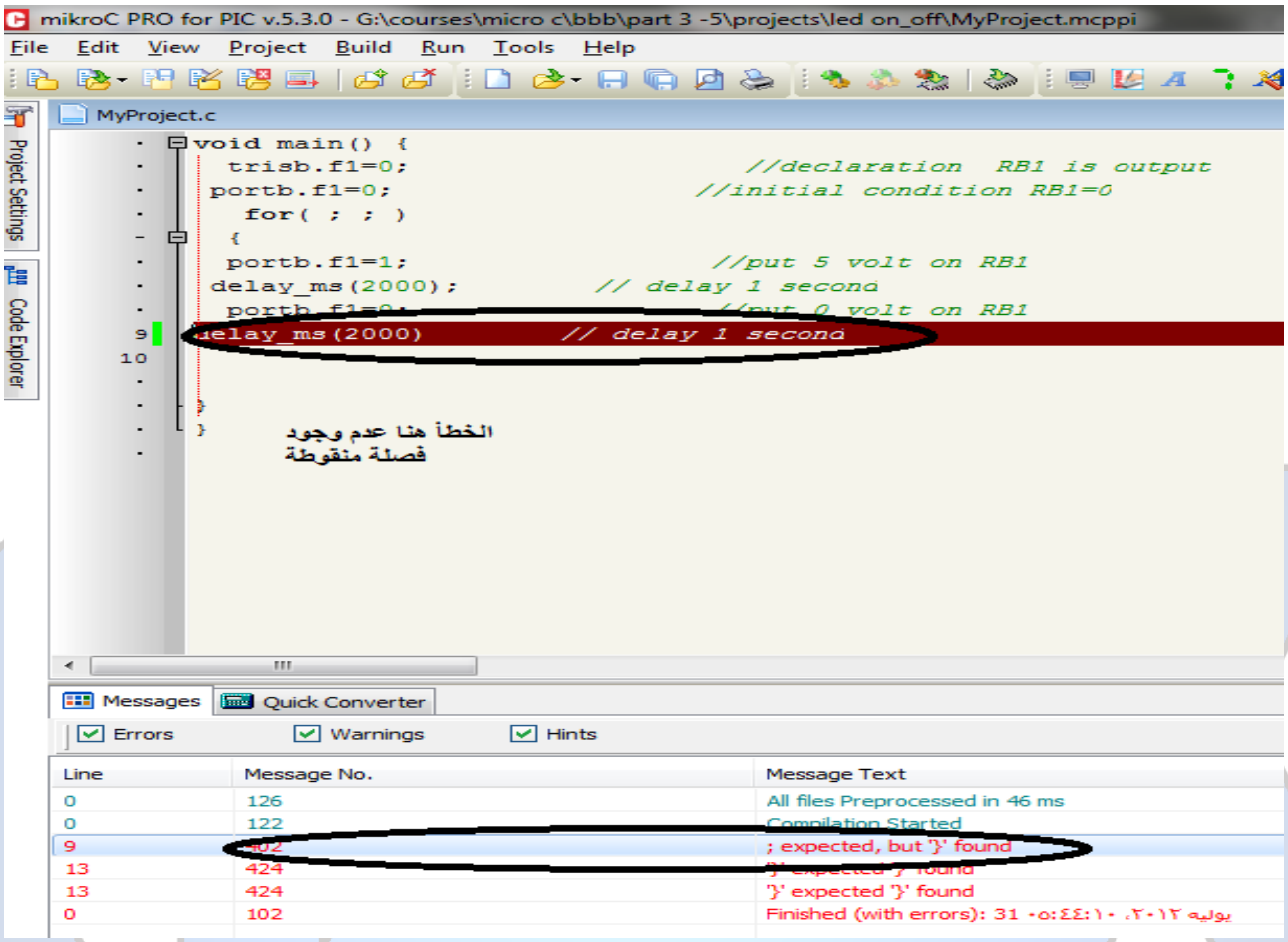


اذا كان البرنامج صحيح

سيكتب لك فى قائمة messages الموجودة بأسفل project linked successfully



اذا كان بالبرنامج اخطاء سيكتب فى النافذة الاخطاء باللون الاحمر اضغط على الخطأ وسيتم توضيحة على نافذة الكود بهذا الشكل



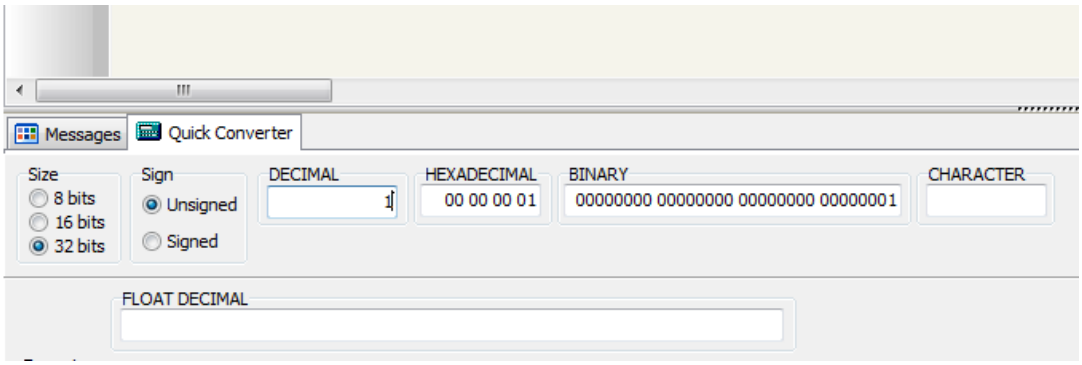
يتم تصحيح الاخطاء وعمل build مرة أخرى الى ان يتم عمل البرنامج بدون أخطاء .

بعد عمل البرنامج اذا أردت ان تفتحة مرة ثانية لتعدل فيه وعمل compile للتعديل الجديد من قائمة project اختار open project .

واذا اردت فتحة لقراءة البرنامج فقط أو اخذ copy لجزء منه دون التعديل فى محتواه يتم فتحه من file ثم اختار open .

بعض الاشياء الهام التعرف عليها فى البرنامج :

1_التبويب الذى بجوار messages يمكن كتابة اى رقم decimal فى الخانة decimal ويتم تحويلها الى binary والى hex decimal كما هو موضح بالشكل



2_ اذا تم فتح برنامج MICROC وكان به برنامج مكتوب سابقا يتم غلق هذا البرنامج اولا قبل انشاء برنامج جديد وذلك من قائمة PROJECT ثم close project سيتم غلق البرنامج المفتوح حاليا ثم ابدء بانشاء برنامج جديد من new project كما وضحنا سابقا .

3_ يتم الضغط على f1 لاستدعاء ال help للبرنامج والذى يحتوى على بعض الاوامر وال libraries والتى سنتحدث عنها لاحقا .

4_ قائمة VIEW لاظهار القوائم الجانبية للبرنامج أو اخفائها والادوات الخاصة بالبرنامج .

C language tutorial

1_ الامر include

#include <.....>

ويستخدم لاستدعاء بعد الدوال والتى تقوم بأداء وظيفة معينة .

2_ الامر define

#define PI 3.14

تخزين قيمة ثابتة فى اسم فبدل من كتابة 3.14 نكتب pi .

3_ الدالة main

فى اى برنامج ميكرو لازم تكون MAIN FUNCTION موجودة لان الميكرو اول ما بيشتغل بينفذ اولا ال code الموجود داخل main

```
Void main () {
.....
.....
}
```

4_ تعريف المتغيرات

Data types

Type	Size (bits)	Range
unsigned char	8	0 to 255
unsigned short int	8	0 to 255
unsigned int	16	0 to 65535
unsigned long int	32	0 to 4294967295
signed char	8	−128 to 127
signed short int	8	−128 to 127
signed int	16	−32768 to 32767
signed long int	32	−2147483648 to 2147483647
float	32	±1.17549435082E-38 to ±6.80564774407E38
double	32	±1.17549435082E-38 to ±6.80564774407E38
long double	32	±1.17549435082E-38 to ±6.80564774407E38

لو ال system اللى محتاج اعرفة ثلاجة هعرفه على انه signed عشان محتاج قيم سالبه لو ال system فرن اكيد هحتاج unsigned لانى مش محتاج قيم سالبة لكن هحتاج درجة حرارة اكتر من 255 لذا هستخدم unsigned

Variable names

1_ variable name consists of (letter, number or underscore)

2_ variable name must begin with a letter or underscore

3_ variable name cannot be a reserved word such as (void, main, while..... for)

4_ c is sensitive for small or capital- -الخاصية دي ممكن تشغيلها أو إيقافها

5_ لازم اضع semi Column ; بعد نهاية الجملة

أنواع المتغيرات

متغير عدد صحيح (int):

int x=10;

متغير عدد كسرى(float) :

float y;

متغير حرفى (char) :

يكتب حرف واحد فقط

char eng ='a';

متغير سلسلة حرفية (string) :

char eng []="engineer";

توضع الجملة بين قوسين تنصيص

5_ comments :

فهى توضح فائدة امر معين بداخل البرنامج حتى يسهل على من يقرأ البرنامج فهم كيفية عمله .

//one line comment

باكتب ال comment فى سطر واحد لو نزلت على سطر تانى هيكون امر جديد .

/* long comment */

اقدر اكتب ال comment فى كذا سطر

function call _6

وهى تستعمل فى عمل برامج صغيرة بداخل البرنامج الاساسى يتم استدعائها كلما احتجت اليها فمثلا لو محتاج فى البرنامج الاساسى انى اجيب مكعب رقم معين اعمل function واسميها مثلا cube واقول

Void cube(int x){

Return x*x*x;

}

لو احتجت مكعب اى رقم انادى على ال function دى وادخلها input وهو الرقم اللى محتاج مكعبة وهى ترجعلى بمكعب الرقم وانادى عليها بهذة الكيفية ;(3) cube طبعا اول ما البرنامج يشوف الامر ده هيروح علطول لل function cube ويعطى لل x القيمة الموجودة بين الاقواس وهى 3 ويرجع بحاصل ضرب 3*3*3 وبعدين يكمل البرنامج عادى .

ممكن اعمل function باسم hello أنادى عليها من داخل main

وممكن اعمل function باسم bye انادى عليها من داخل hello

ال sequence اللى بيمشى بيه البرنامج كالاتى :

يبدأ البرنامج بتنفيذ main يمشى فى تنفيذها لحد ما يلاقى ;()hello يروح ل hello يمشى فيها لحد ما يقابل ;()bye يروح ل bye يخلصها ثم يرجع يكمل hello ثم يرجع يكمل main .

طب ينفع افضل انادى على function من جوه function علطول ولا ليها limit معين ؟؟؟؟

لازم لما انادى على function من جوه function لازم احفظ مسار الرجوع فى ال stack بداخل الميكرو واللى حجمه 8 level لو زودت عن 8 level هيكون ال stack over flow وقتها الميكرو هيخرف لانه مش فاكر مسار الرجوع .

الخلاصة مش المفروض اعمل اكتر من 8 listed call , ممكن انادى ال function الواحده كذا مرة من جوه function تانيه لكن مش ينفع انادى function من جوه function اكتر من 8 مرات .

Variable scope:

1) Local

يعنى معرف بداخل ال function دى مش فى اى function تانية .

2) Global

يعنى معرف outside ال function كلها وبيكون معروف لكل ال functions .

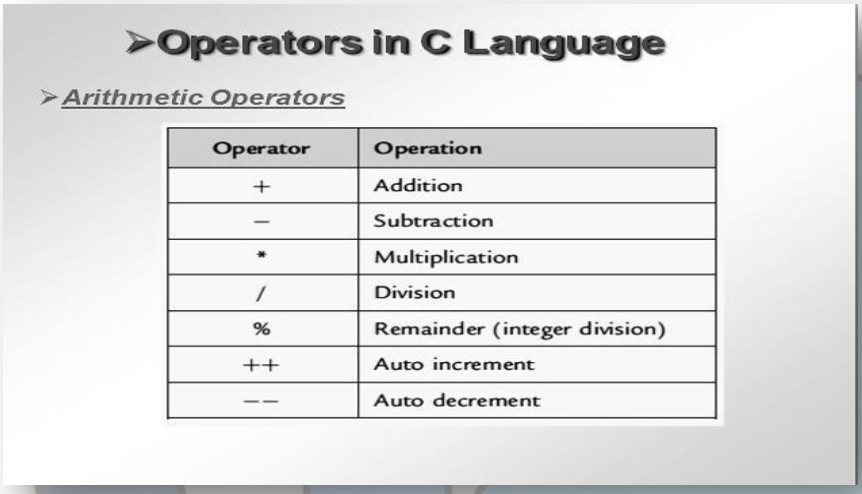
طب ما اخلى كله global احسن هخسر اية ؟؟؟

الحقيقة ان لما أخلى ال local variable فان المتغير بيموت بمجرد أن ال function الخاصة بيه تنتهى يعنى ممكن يجى variable مكانه

اما لو global مكانه بيتحجز مش ممكن يجى variable يحل مكانه فالافضل ان المتغير لو ينفع يكون local نخليه كده لان ال RAM اللى عندى صغيرة

Operators:

1) Arithmetic



لو مثلا عندى برنامج صغير
كالاتى :

Unsigned char x=5, y= 2,q;

q=x/y;

يساوى هنا تعنى assignment operator فلازم يكون قبلها متغير يعنى حاصل قسمه X/Y بيكون included داخل q فالناتج q=2 لانه فى ال unsigned مفيش fraction كسور باخذ الرقم الصحيح .

q=x % y;

5/2= 2, 1/2

باقى القسمة 1 يعنى q=1

x++;

x=6

y--;

y=1

لاحظ

لو عندى برنامج

Unsigned char x;

Unsigned int y;

y=100;

x=y;

ازاى احط 16 bit فى 8 bit الميكرو بياخد ال 8 ويرمى الباقى

If y=256;

x=y ;

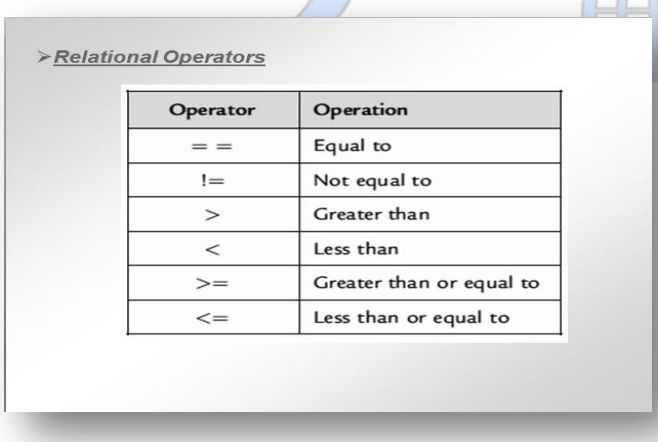
فإن

x=0

بما ان ال y =0000000010000000 فان الميكرو هياخد ال 8bit ويرمى الباقى يعنى x=0 .

احيانا الشخص المبرمج بيكون قاصد كده عشان لو عندى شاشة LCD مش بتشيل غير 8 bit ابعت الرقم على مرتين .

2) Relational :



اذا كان e=2 w=4 وقال ان q=w>e فان q=1 لان الشرط حقيقى .

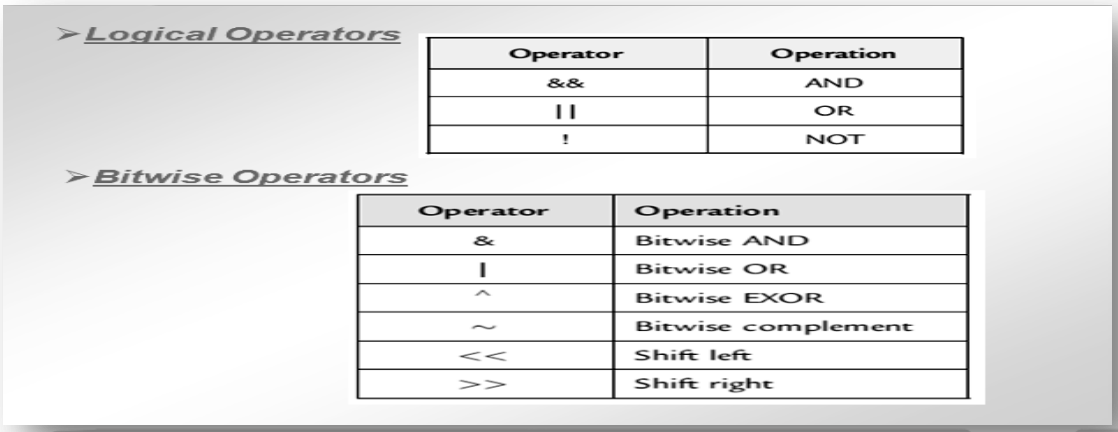
One mean true zero mean false

q=x==y

يساوى الاولى هى assignment operator اما الثانية هى relational operator تستخدم لمقارنة رقمين هل هما متساويان أم لا .

فالشرط غير صحيح اذا Z=0 .

3) Logic operators



➢ *Logical Operators*

Operator	Operation
&&	AND
\|\|	OR
!	NOT

➢ *Bitwise Operators*

Operator	Operation
&	Bitwise AND
\|	Bitwise OR
^	Bitwise EXOR
~	Bitwise complement
<<	Shift left
>>	Shift right

ال bit wise بيتعاملوا مع ال variable على مستوى ال bit .

Ex:

Unsigned char x=7, y=27, q;

لاحظ ان اى nonzero value تعتبر 1

q=x&&y;

q=1;

X=00000111

Y=00011011

q=x & y;

يحول كل منهم ال binary ويشوف كل bit قصاد التانى

q=00000011=3

q=3

Ex:

X=255;

Y= ~x;

X=11111111

Y=00000000

Y=0

Ex:

X=4

Y=x<<2

2 number of shift

بيزيح خانتين من ال 8 ويضع مكانهم اصفار

x=00000100

y=00010000

y=16

One shift left as u multiply by 2

Y=4*2*2=16

Statements

IF FOR WHILE

If:

If (expression) {

.....

.....

}

لو كان الناتج لل expression يساوى 1 ينفذ الجملة بين ال curl brackets اما لو zero يعدى ال {} ويكمل عادى.

If (expression) {

.....

.....

}

Else {

.....

.....

}

لو true ينفذ اللى بين ال curl brackets الخاص ب if لو false ينفذ اللى بين ال curl brackets الخاص ب else

If (expression) {

.....

.....

}

Else if {

.....

.....

}

Else if {

.....

.....

}

Else {

.....

.....

}

ينفذ اللى بين ال curl brackets الخاص ب if اذا كان true فاذا كان false يروح ل else if الاولى اذا كان ال expression اللى بين ال curl brackets الخاص ب else if الاولى true ينفذه اذا كان false يروح ل else if التانية اذا كان حقيقى ينفذه لو خطأ ينفذ اللى بين ال curl brackets الخاص ب else .

Else is optional يعنى لو مش كان موجود اى حالة من الحالات السابقة مش تعمل حاجه.

لاحظ

لو جملة واحدة مع if مش شرط اضع ال {} اما لو كذا جملة يعنى مثلا لو المفتاح مفتوح الليد تنور والموتور يتحرك دا جملتين وقتها لازم اضع ال {} لو مش وضعتها هينفذ الامر الاول بس.

For:

for(Initialization; condition; increment){

.....}

Unsigned char I;

For (i=0; I<10; I++) {

.....}

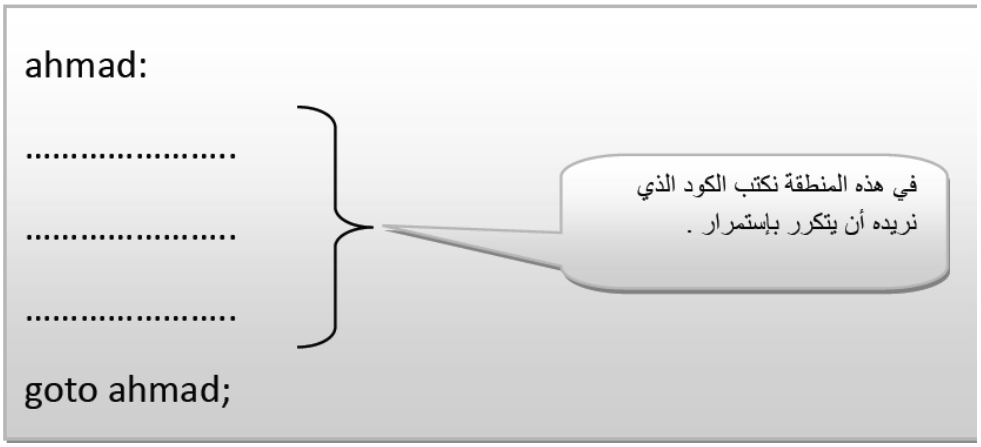
ال sequence اللى بيمشى بيه البرنامج عند I=0 دى القيمة الابتدائية اللى بتبدأ عندها ال loop تتنفذ يدخل على ال condition يشوف هل 0<10 اه يبدأ ينفذ البرنامج اللى بين {} ثم يرجع لل increment هل 1<10 نعم يبدأ ينفذ البرنامج اللى بين {} تانى وهكذا لحد ما ال condition مش يتحقق وبكده ال loop تقف .

For (;;){

.....}

دى infinite loop عدد لا نهائى

يفضل استخدام for(;;) لان الميكرو بعد نهاية main بيقف فهضطر ان اعمل restart للبرنامج



عشان يحمل من جديد اما ال infinite loop عمرى ما هوصل لنهاية main وبكده ال system مش هيقف .

شكل اخر لأشكال ال infinite loop

While:

while (exp) {

.....}

طول ما ال expression is true يفضل ينفذ البرنامج لو false مش يعمل حاجة يمكن استبدال while ب for

For (; exp ;){.....}

While (1) تكافئ for (;;) بمعنى انها loop لا نهائية .

Do while:

Do {

.....}

While (condition)

تقوم بتنفيذ البرنامج مرة واحدة ع الاقل ثم تنظر الى الشرط هل هو true or false فاذا كان false مش تعمل حاجة لو true ترجع ل do تنفذ ال statement اللى بداخلها مرة تانيه وهكذا

Body of the program

اولا : تعريف الثوابت

ثانيا : دالة MAIN

ثالثا : الاعلان عن المتغيرات اسمائها ونوعها

رابعا : الداتا احدد اتجاهها هل هى input أم output .

خامسا : لابد ان اضع قيمة ابتدائية للمتغيرات .

سادسا : infinite loop باستخدمها عشان طول ما الميكرو موضوع فى الدائرة فالبرنامج شغال مش

```
#define  PI   3.14

void main( )
{
    Variable declaration
    Data direction
    Initial values
while(1)
  {
        .....;
        .....;
        .....;
  }
}
```

بيتنفذ مرة واحدة ويقف بل يتنفذ باستمرار طول ما ال power موجودة .

I/O ports

Data direction

وبحدد هل ال pin تعمل ك input أم ك output .

output _____ 0

input _____ 1

ويتم ذلك عن طريق كتابة هذا الامر فى code ال C :

TRISX=Value;

X: - is the port name we want to control its pins [A, B, C, D, and E]

VALUE :- it has one of two values [0 , 1] zero for make the pin output and one for making the pin input .The value may binary or hexadecimal or decimal

TRISX=0b10110101; //Binary so we use 0b

وهذا معناه أن pin0 فى PORTB تعمل ك input

1Pin فى PORTB تعمل ك output.....وهكذا.

لو عاوز اشتغل على PIN معينه بداخل ال PORT مش ال PORT كله استخدم هذا الامر :

TRISX.fN = 0 or 1;

حيث N هو رقم ال PIN بداخل ال PORT فلو عاوز PIN3 بداخل PORTB تشتغل OUT فيكتب الامر كالاتى :

TRISB.f3 = 0;

ملحوظة :

الامر TRIS أو PORT من الممكن كتبتهم SMALL OR CAPITAL LETTER.

Examples

Make port B output	Make port C input
`Trisb = 0b00000000 ;`	`TRISC = 0b11111111 ;`
Make pins 1 , 3 , 5 in port B input	Make pins 0 , 3 , 7 in port C output
`Trisb = 0b00101010 ;`	`Trisb = 0b01110110 ;`

ال default ان كل pin عبارة عن input، ال pin لو input ومش واصل عليها حاجة وقست عليها بتكون float او high impedance لانى مستنى ال input

ال pin لو OUT ومش موصل عليها حاجه تعطى 1 or 0 نتيجة random ومن الافضل عدم الاعتماد على ال default وتعريف كل شئ لضمان جودة ال system .

PIN OUTPUT

بحدد خرج ال PIN هل هو 0V أو 5V .

اما منا الان سجل عمل PORTB فهو وببساطه عن موقع في ذاكرة التشيب او بايت مؤلف من ثماني بيت فكل ما يكتب في سجل العمل هذا تظهر نتيجته في البين التابع له .فاذا كتبنا 1 في البيت RP3 من سجل العمل ستكون كهربا البين نفسه +5 فولت و هذا يعني 1 والعكس صحيح .

سجل عمل خاص

PORT B

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
1	0	0	1	1	0	1	1

ويتم ذلك عن طريق الامر :

PORTX = Value;

put 5 volt on all pins of port B ***PORTB = 0b11111111 ;***	put 0 volt on all pins of port C ***PORTC = 0b00000000 ;***
put 5 volt on pins 1 , 3 , 5 of port B And other pins are 0 volt ***PORTB = 0b00101010 ;***	put 5 volt on pins 0 , 2 , 4 of port C And other pins are 0 volt ***PORTC = 0b00010101 ;***

لو عاوز PIN واحدة من PINS ال PORT اضع عليها اما 0V / 5V استخدم الامر الاتى :

PORTX .fN =0 or 1;

Make the pin 3 on port b as input ***TRISB . f3 = 1 ;***	put 5 volt on pin 2 of port C ***PORTC . f2 = 1 ;***
Make the pin 5 on port c as output ***TRISC . f5 = 0 ;***	Put 0 volt on pin 7 of port B ***PORTB . f7 = 0 ;***

كيفية استقبال قيم INPUT من الميكرو :

Receive values from microcontroller pins :

الطريقة الاولى :

يقوم البرنامج بقراءة قيم ال INPUT والتى سنتكلم عن كيفية دخولها فى الدرس القادم ويقارنها بالقيمة 0b11111111 فاذا كانت تتطابقها يقوم باخراج هذة القيم 0b00000000 على PORTC اذا لم تطابقها يقوم باخراج القيم التالية 0b11111111 وهذا كمثال توضيحى فقط .

```
If  (PORTB = 0b11111111)
  {
    PORTC = 0b00000000 ;
  }
else
  {
    PORTC = 0b11111111 ;

  } …
```

```
void main( )
{
unsigned char eng ;
TRISA = 0b00011111 ;
TRISB = 0 ;
 PORTB  = 0 ;
while(1)
{
    eng = PORTA ;
    portb = eng ;
}
}
```

الطريقة التانية :

يقوم بتعريف PORTB على انه OUTPUT ويجعل خرجه فى البداية ZERO ويقوم بالاعلان عن متغير ENG ويقوم بقراءة ال INPUT من على PORTA ويضعها فى ENG ثم يخرجها على PORTB .

الأخطاء في لغة الـ C

رسائل الأخطاء الشائعة ومعناها

الرسالة	المعنى
Array size too large	حجم المصفوفة كبير
Bad file name format in include directive	اسم المكتبة خطأ
Case outside of switch	جمله case خارج جمله switch
Compound statement missing{	القوس } مفقود
Conflicting type modifiers	تعارض في النوع
Could not find file 'filename'	اسم المكتبة غير موجود
Declaration missing;	إعلان مفقود
Declaration syntax error Suspicious pointer conversion	خطأ في بناء الجملة نقطة تحول مريبه
Declaration terminated incorrectly	انتهاء خاطئ للإعلان
Default outside of switch	جمله Default خارج switch
Default value missing	القيمة مفقودة
Division by zero	قسمه على صفر
do statement must have while	جمله do تنقصها جملة while
do-while statement missing)	جمله do-whileتنقصها (
do-while statement missing(	جمله do-whileتنقصها)(
do-while statement missing;	جمله do-whileتنقصها ;
Duplicate case	مكررة
expected)	عدم وجود قوس (
expected(	عدم وجود قوس)
expected,	عدم وجود فاصلة
expected}	عدم وجود قوس {
expected{	عدم وجود قوس}
expected>	عدم وجود قوس <
Expression syntax	جملة غير صحيحة
For statement missing)	جمله for تنقصها قوس (
For statement missing(	جمله for تنقصها قوس)
For statement missing;	جمله for تنقصها الفاصلة المنقوطة ;
Function call missing(	الدالة تنقصها قوس)
function' cannot return a value'	الدالة لا تستطيع الرجوع بقيمة
function' must be declared with no parameters'	يجب أن تعلن الدالة بدون بارميتر
function' must be declared with one parameter'	يجب أن تعلن الدالة بباراميتر واحد
function' must be declared with two parameters'	يجب أن تعلن الدالة واثنان بارميتر
If statement missing)	الدالة تنقصها قوس (
If statement missing(	جمله if تنقصها قوس)
Misplaced else	جمله if تنقصها قوس)
Undefined symbol	متغير غير معرف
Unable to open include file ' '	لايمكن فتح الملف(المكتبة)
Statement massing ;	عدم وجود فاصله ; في السطر السابق

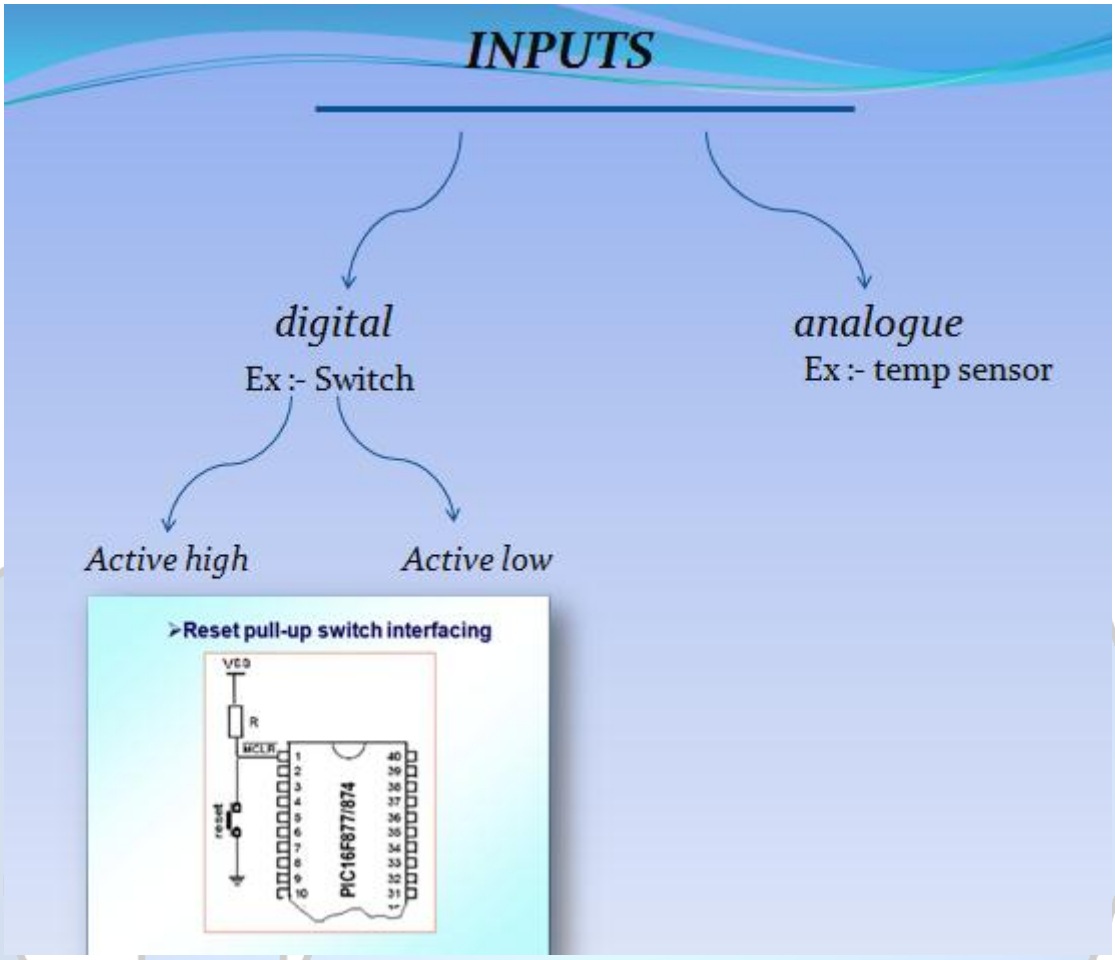
Chapter (3)

Input & output devices



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

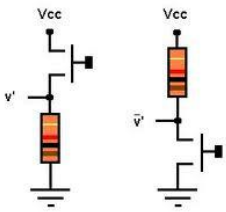
Input & output devices:



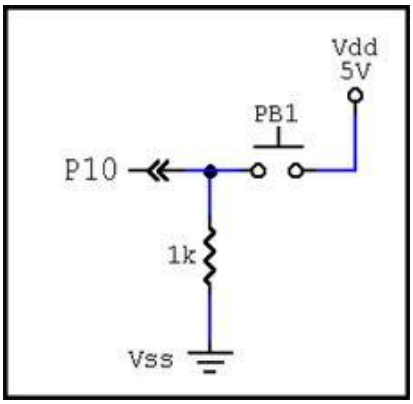
1) Digital input devices

A) Push buttons





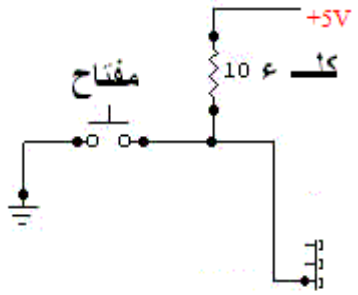
يوجد طريقتين لتوصيل ال push button :



الطريقة الاولى (active high switch) :

If pin is high button is pressed else button is released

الطريقة الثانية (active low switch):



If pin is low button is pressed else button is released

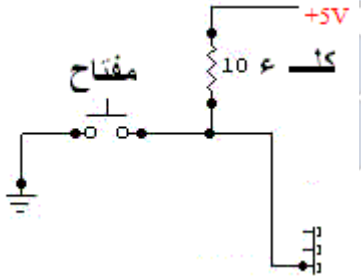
يفضل استخدام الطريقة الثانية لان احتمال ان ال VCC تعطى low قليل فيفضل على ال pin high يعنى مفيش reset .

2) Limit switch





زى اللى موجود فى الاسانسير عبارة عن ذراع فى اخرة عجلة عند مرور الاسانسير عليه يقفل ال switch وبكده اكون عرفت ان ده الدور اللى مفروض انزل فيه حيث يكون ال switch اللى فى الدور اللى مفروض انزل فيه مغلق وال switch فى باقى الادوار مفتوح .



أما فى الميكرو فانه يعامل معاملة ال Push button

رمز ال limit switch يختلف عن ال push button لكن يوصل المفتاح كما هو موجود بالشكل .

3) Proximity switch



القطعه البلاستيك فى المقدمة تعمل ك sensor بيحس بأى شئ امامه .

يوجد منه نوعان :

- First type (inductive)

 بيحس بال metal

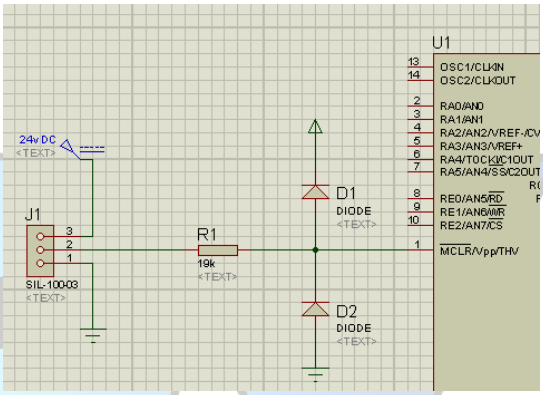
- Second type (capacitive)

بيحس بال metal& nonmetal

لو فيه حاجه قدامه الخرج بيكون high 24v DC

لو مش فيه حاجه قدامه الخرج بيكون low 0V

طب الميكرو بيشتغل ب 5 فولت وليس 24 فولت لذا لازم اعمل clamping لل 24فولت وتكون دائرتها بهذا الشكل:



لو واصل zero من ال sensor معنى كده ان D1& D2 Is off يعنى O.C يعنى واصل لل pin zero volt

أما لو داخل من ال sensor 24v فان

D1 is on و

D2 is off يعنى داخل لل pin 5 فولت لان الدايود عمل S.C بين ال VCC وال PIN ويكون فى فرق جهد ع المقاومة 24-5=19 فولت .

فى هذة الدائرة D2 مش له لازمة الا لو الفولت اللى داخل سالب

المفاجأة ان بداخل الميكرو دائرة clamping جاهزة يعنى مش هحتاج اضع دايود لكن لازم اضع المقاومة عشان تشيل فرق الجهد بين ال 24 فولت وال 5 فولت

طب المقاومة دى قيمتها كام ؟؟؟؟

لو فرضنا مر فى المقاومة 1mA لانى بيخرج من ال switch اقصى تيار 20mA اذا

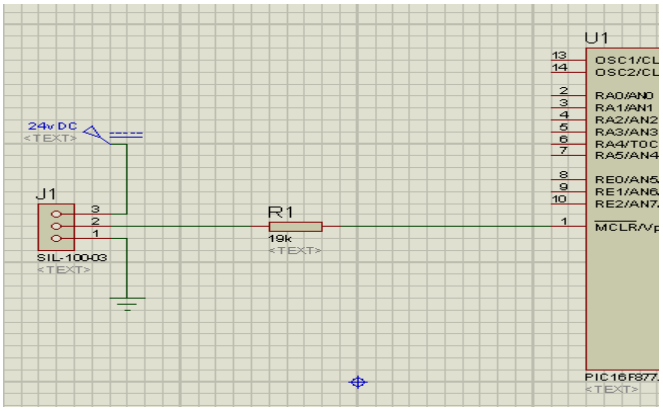
R= (24-5)/1mA=19KΩ

تتعين المقاومة بقياس الباور ليها

P = I^2R = (1mA)^2*19K=19mW

المقاومة اللى بشتريها من السوق بتكون ربع وات يعنى شغاله معايا .

الدائرة تكون بهذا الشكل

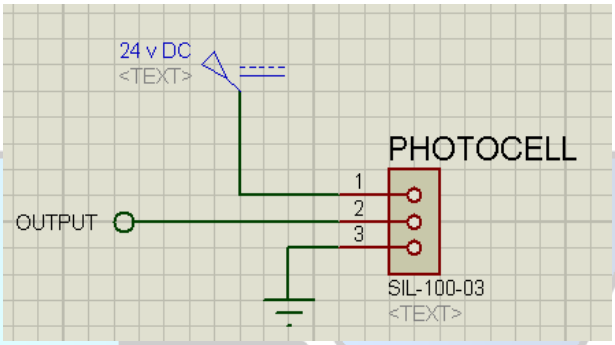


4) Photocell

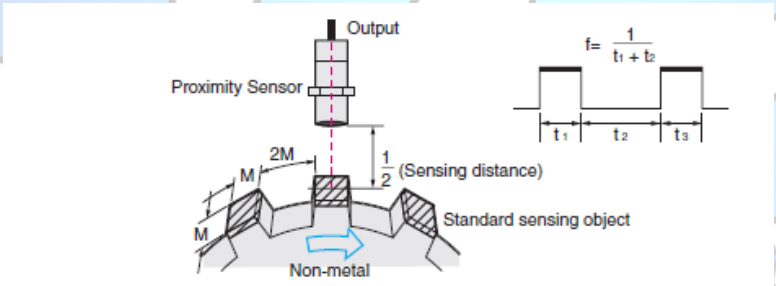


لو حصل ان فيه object فى النص بين ال sensor وال reflector الضوء مش هينعكس يعنى الخرج high لو مفيش objectالضوء هينعكس فالخرج يكون low حيث ان ال SENSOR يخرج اشعه تحت الحمراء .

يوصل بهذا الشكل:



ميزة ال photocell عن ال proximity ان المسافة بين ال object وال sensor تكون بالامتار اما فى ال proximity المسافه بالملى متر لكن ميزة ال proximity ان فى الصناعه يوجد dust فممكن ال reflector يتغطى بالتراب فال sensor يعطى high علطول وبكده هحتاج انى انظف ال reflector بانتظام.



5) PIR sensor

Passive infra red

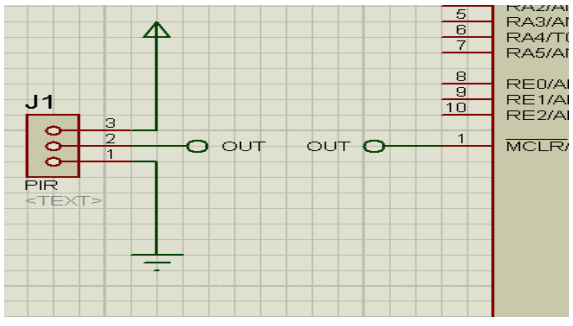


من أشهر الطرق اللى بقدر اعرف بيها فى حد فى المكان ولا لا حيث يوضع ال PIR مثبت على سطح الغرفة وبيشتغل هذا

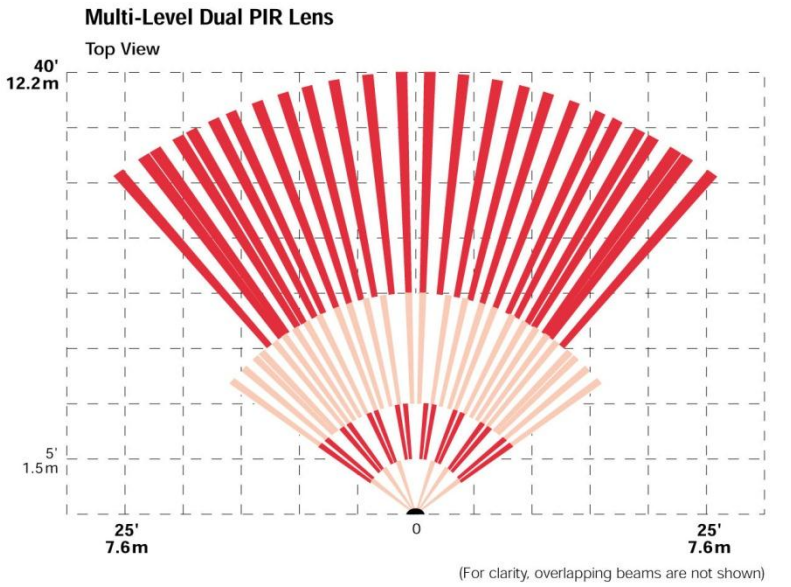


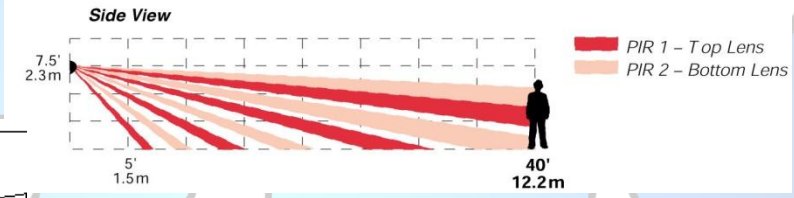
ال sensor ب 5 فولت فقط .

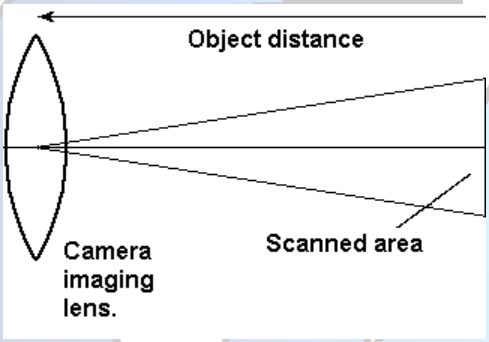
ويوصل بهذا الشكل



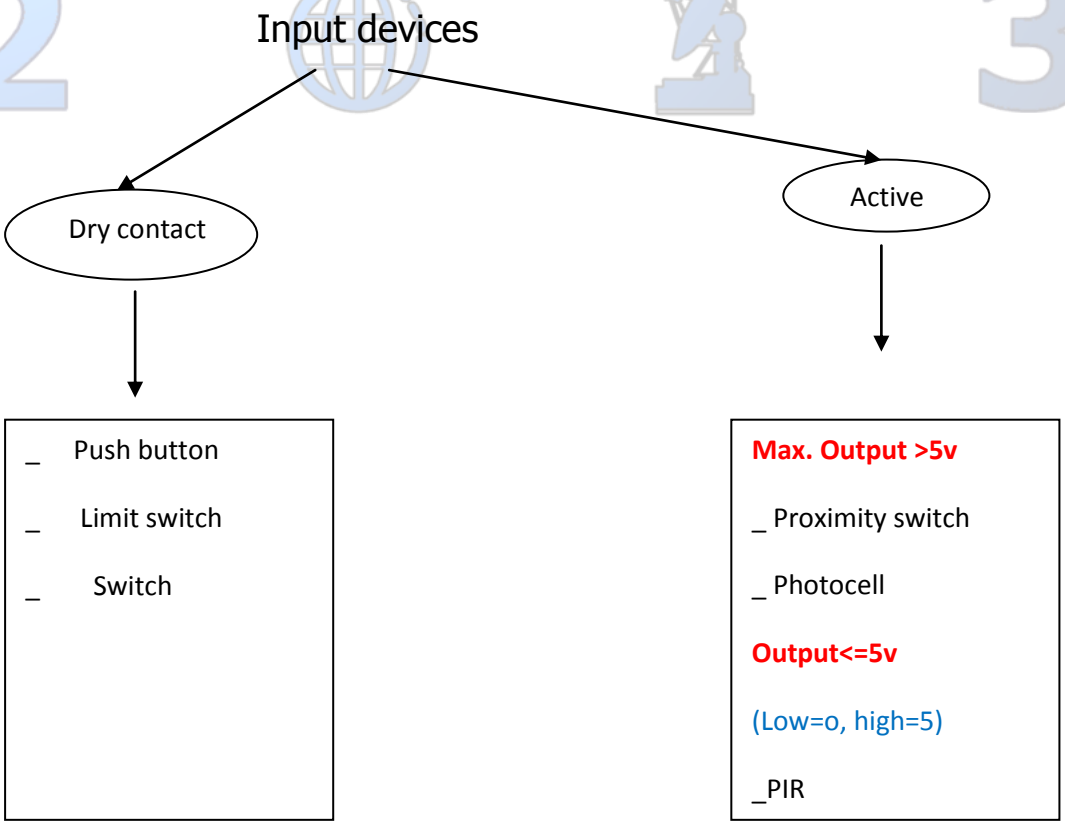
يوضع ال PIR فى اعلى سطح الغرفه ويكون معلوم عندى طول السطح الغرفة وزاوية ميل الاشعة النافذة منه فأقدر أحدد ال مساحة اللى ال sensor بيتفقد فيها .

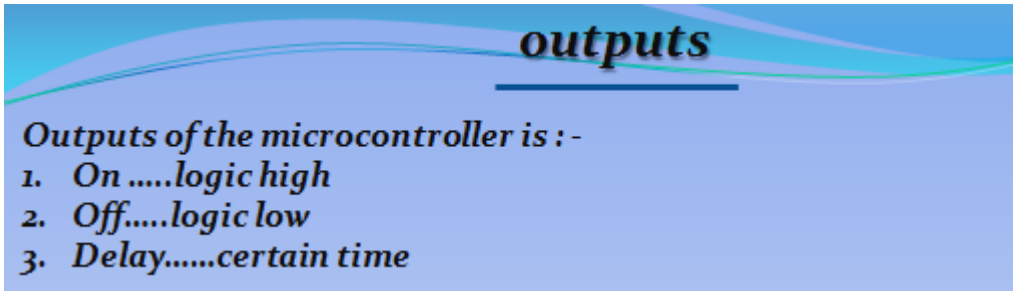






ملخص ما سبق :





ال delay معناه انى لو مخرج على ال pin 5 فولت أقول للميكرو خلى ال 5v على ال pin للمدة دى من الزمن .

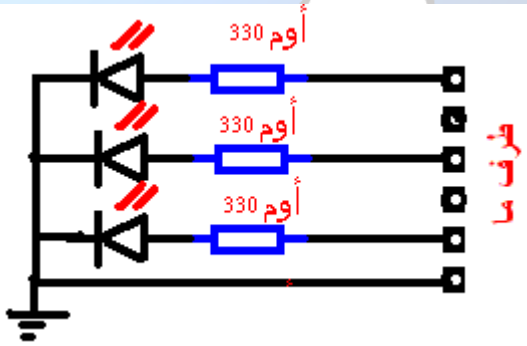
1) Led:

ممكن اختبر الخرج ب led حيث ان ال pin بيخرج منها تيار تقريبا 20mA فلو الخرج high led is on/ low led is off تقريبا لو قرينا ال data sheet لل led هاجد ان ال led بيقع عليها voltage drop قيمته 1.2v لذا يوضع مقاومة مع الليد تشيل فرق الجهد بين ال 5 فولت وال 1.2 قيمة هذة المقاومة ، أقصى تيار خارج من ال pin 25ملى امبير هى :

R= (5-1.2)/25mA=152Ω

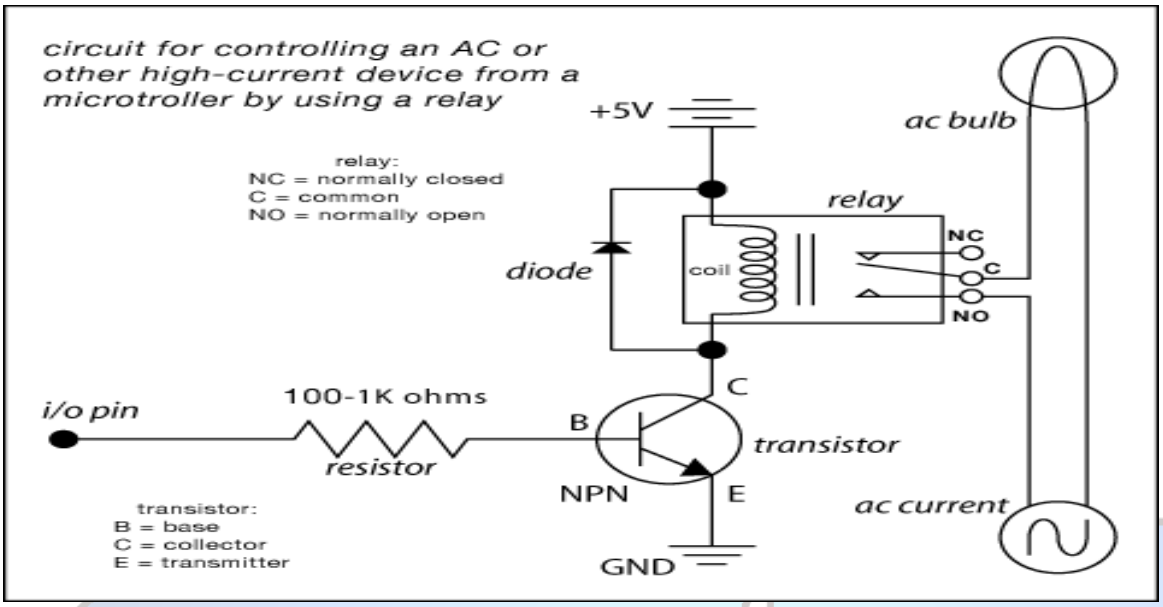
إذن فإن أقل مقاومة يمكن استخدامها بحيث لا تؤثر على ال pin هي Ω152 ويفضل أن تكون قيمة المقاومة أكبر من ذلك لكي لا يتم استهلاك أقصى قيمة للتيار من ال pin

لذا الموجود فى السوق المقاومة220 اوم لذا اضعها قبل الليد



ال device الوحيد اللى مش بيحتاج دائرة interface بينه وبين الميكرو هو الليد لانه بيشتغل بتيار صغير لكن لو انا عاوز اشغل device ب 220v لكن الميكرو أما بيخرج 5v / 0v فى الوقت دا اعمل دايرة interface تربط بين ال 5v الخارجة من الميكرو وال 220v اللى محتاج اشغل بيها ال device لذا اخرج ال 5v من الميكرو اشغل بيهم دايرة ال interface لما يجيها من الميكرو 0v تفصل ال 220v لما يجيها 5v توصل ال 220 v .

2) Interface relay:



يخرج من ال pin تيار صغير عندما يمر تيار فى الملف يتولد magnetic wave يجذب ال contact الخاص بال switch الموجود بال relay وبذلك يكون ال switch closed فيغلق الدائرة فيمر v220 طب ال relay محتاج 30ملى امبير وايضا

12 فولت او 6 او 34 او 22 فولت عشان يشتغل حسب نوعه لكن انا هنا بشتغل ب 12 فولت relay لكن الميكرو يعطى اقصى تيار 20 ملى امبير لذا اوصل على ال pin للميكرو مقاومة وترانزستور NPN ممكن bc547 حيث يدخل من ال base تيار صغير اقدراخده من الميكرو ويمر فى ال collector تيار أكبر يلائم تشغيل ال relay

لما تكون ال pin high يمر تيار فى الترانزستورهيصل الترانزستور الى حالة ال saturation وهيكون بين ال emitter وال collector عبارة عن short circuit فهيوصل ارضى للدائرة فيمر تيار فى ملف ال relay يجذب ال contact وبالتالى يصل لل relay 12 فولت فيشتغل ويمر تيار فى اللمبه فتنور اللمبه , عندما يكون الخرج low فان الترانزستورسيصل الى مرحلة ال cutoff وهيكون بين ال emitter وال collector عبارة عن High impedance يعني Open circuitفلا يصل ارضى للدائرة فلا يمر تيار فى الملف فلا تضئ اللمبة .

اضع مع ال relay دايود 1N4007 وذلك لان الملف بيخزن energy يعنى لما اعمل off هيكون هناك Reverse current ممكن يحرق الترانزستور لما يمر فيه لذا لابد ان اجد له مسار بديل هو الدايود حيث انه فى الوضع الطبيعى عندما تكون الدائرة on فان الدايود يكون 0ff وعندما نغلق الدائرة يكون التيار الراجع سالب فيمر فى الدايود ولا يمر فى الترانزستور .

المقاومة اللى مع الترانزستور تكون 1kΩ .

Chapter (4)

Projects



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

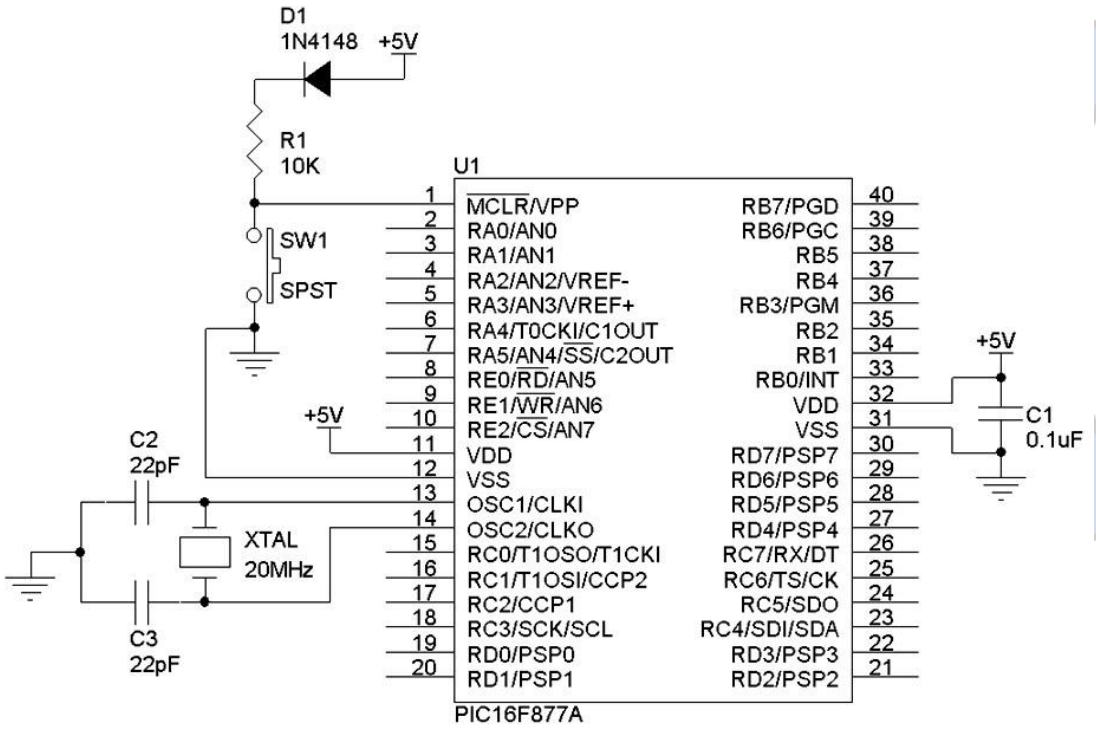
Small project

سنقوم بعمل برنامج صغير كمدخل للتعرف على برمجة الميكرو وعلى كيفية التعامل مع برنامج بروتس وكيفية حرق البيك

البرنامج:

فكرة البرنامج سنقوم بعمل برنامج يقومم باضاءة led لمدة ثانية واطفاءها لمدة ثانية وهكذا .

فى حالة التوصيل ال hardware هتكون التوصيلة التالية ثابته عندى مع الاتفاق كما ذكرنا سابقا ان pin1 موصل عليها فقط مقاومة 10k مع ال VCC والكريستالة اللى هنشتغل عليها 4MHZ .

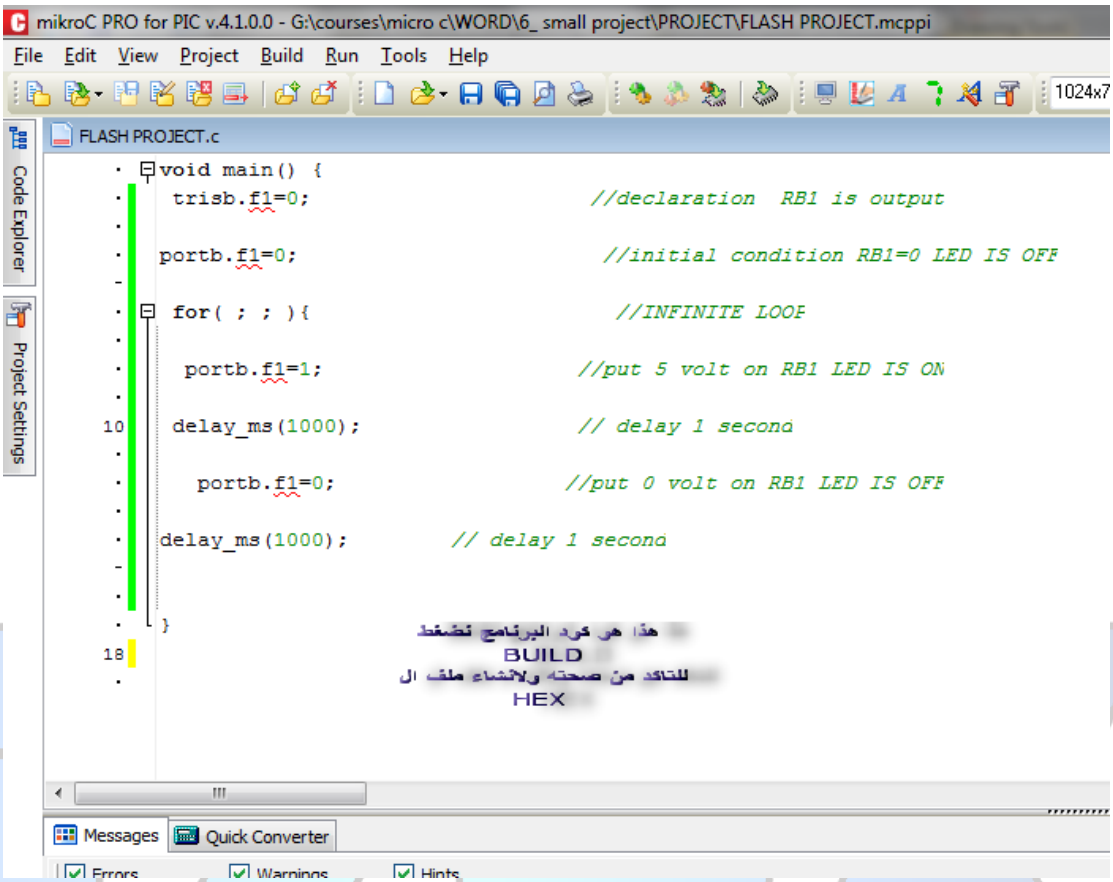


أولا : نقوم بتحديد ال INPUT وال OUTPUT

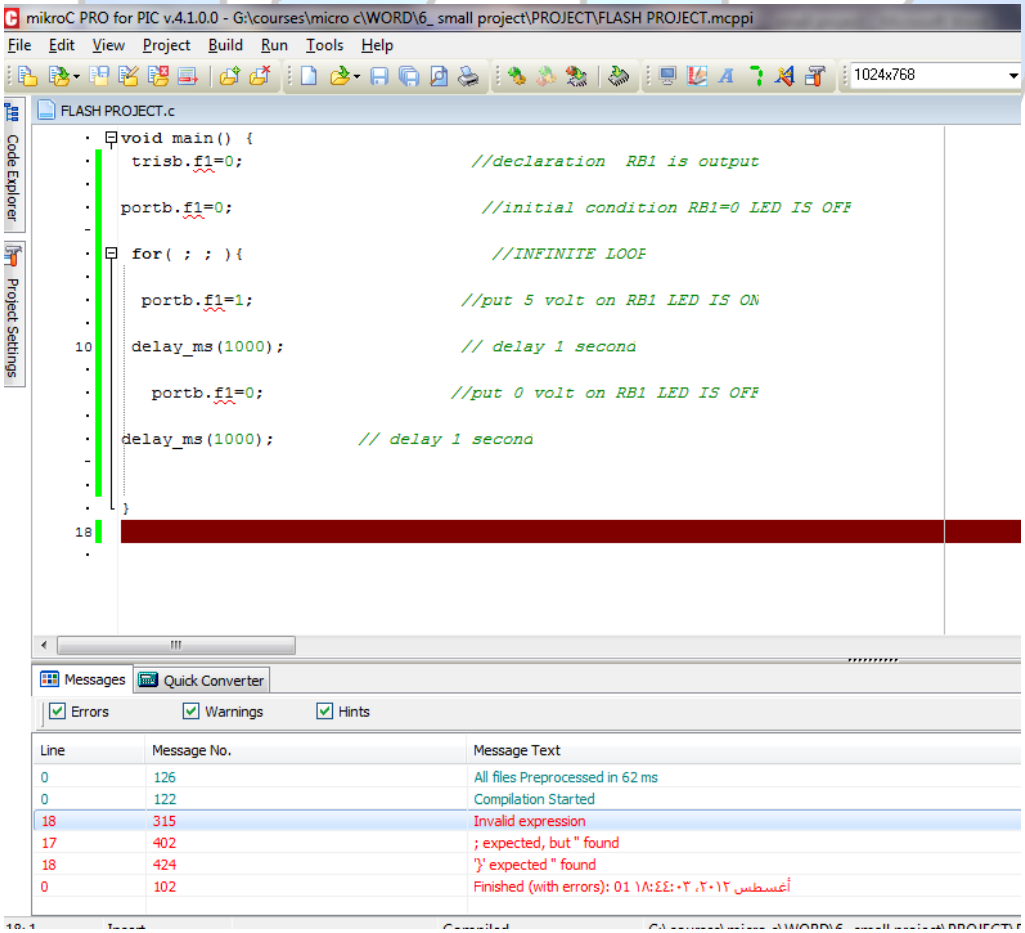
عندى خرج واحد فقط

ثانيا : كتابة البرنامج المطلوب :

بعد انشاء NEW PROJECT نقوم بكتابة البرنامج

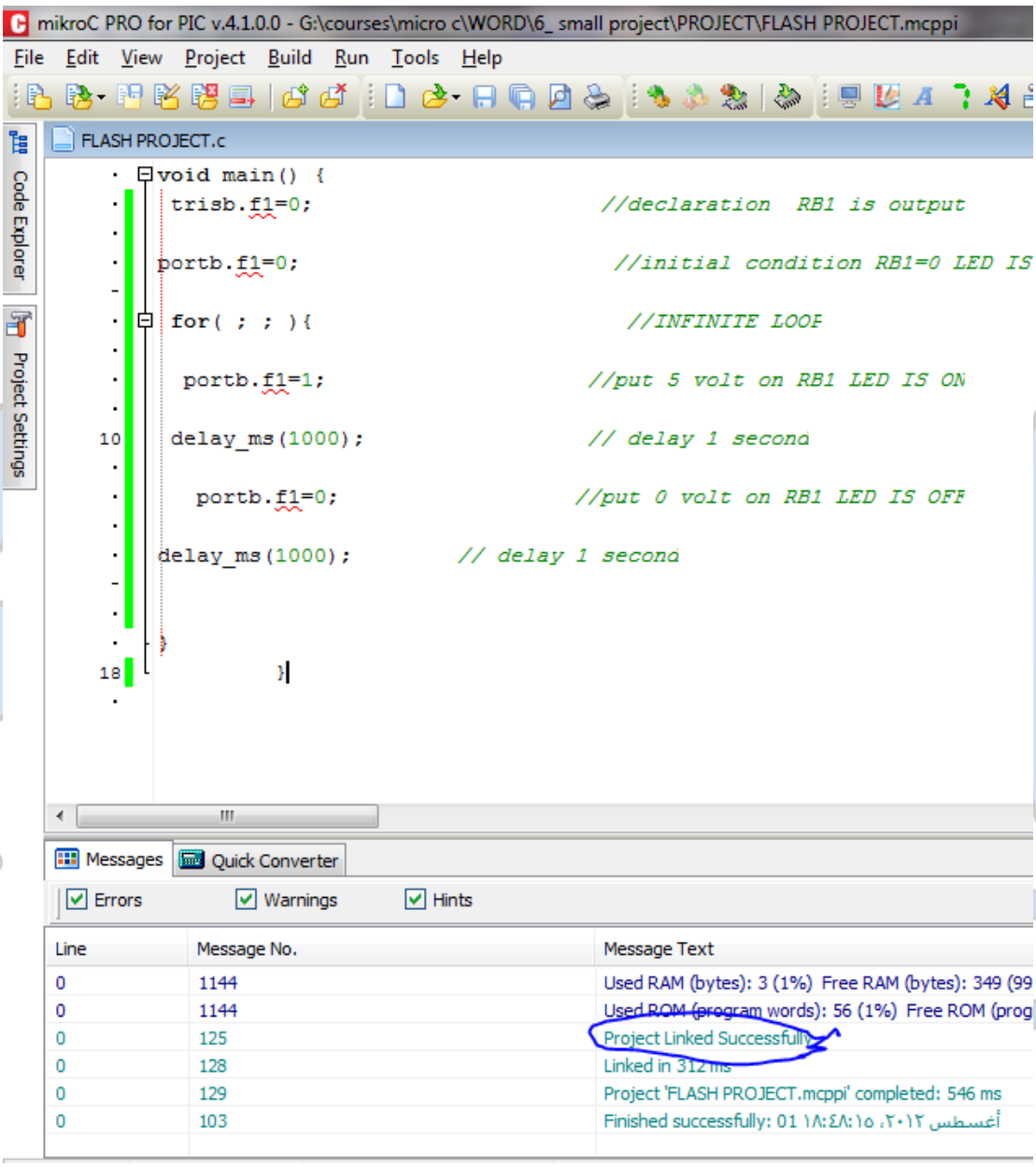


لقد تم كتابة الكود بشكل خاطئ وتم ظهور ذلك فى رسالة أسفل كما بالشكل



قم بالضغط على الخطأ ومراجعة الكود وتصحيح الكود مرة أخرى فنجد انه كان ناقص { لانى محتاج اقفل قوس FOR وايضا قوس MAIN .

بعد تصحيح الخطأ ستظهر هذة الرسالة ليكون كود البرنامج النهائى بهذا الشكل :



دلوقت الميكرو هينور ويطفى الليد زمن ال flashing سريع جدا العين مش هتلاحظه

عاوز اعمل delay عشان اقدر اشوف الليد وهى بتنور وتطفى فالميكرو سى بيوفر ليه function باسم delay بتجعل الخرج مستمر على رجل الميكرو الفترة اللى انا عاوزها وبكده اقدر اشوف الليد وهى بتنور وبتطفى .

نقوم بعمل simulation للدائرة على برنامج بروتس مع العلم ان التوصيلات الثابته للميكرو والمرسومة بالرسمه اعلاه لا يشترط وضعها فى البروتس والبرنامج هيشتغل سليم باذن الله .

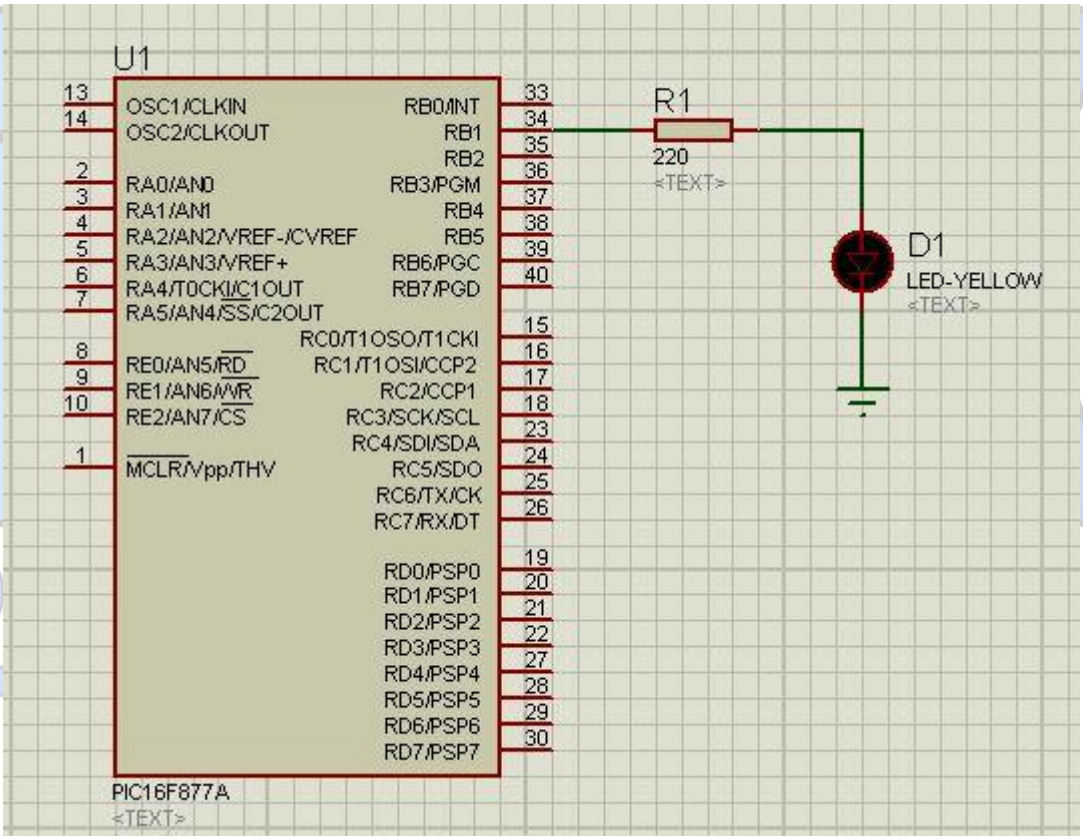
Explain of proteus program

ممكن اختبر الخرج ب led حيث ان ال pin بيخرج منها تيار تقريبا 20mA فلو الخرج high led is on/ low led is off تقريبا لو قرينا ال data sheet لل led هاجد ان ال led بيقع عليها voltage drop تقريبا قيمته 1.2v لذا يوضع مقاومة مع الليد تشيل فرق الجهد بين ال 5 فولت وال 1.2 قيمة هذة المقاومة ، أقصى تيار خارج من ال pin 25ملى امبير هى :

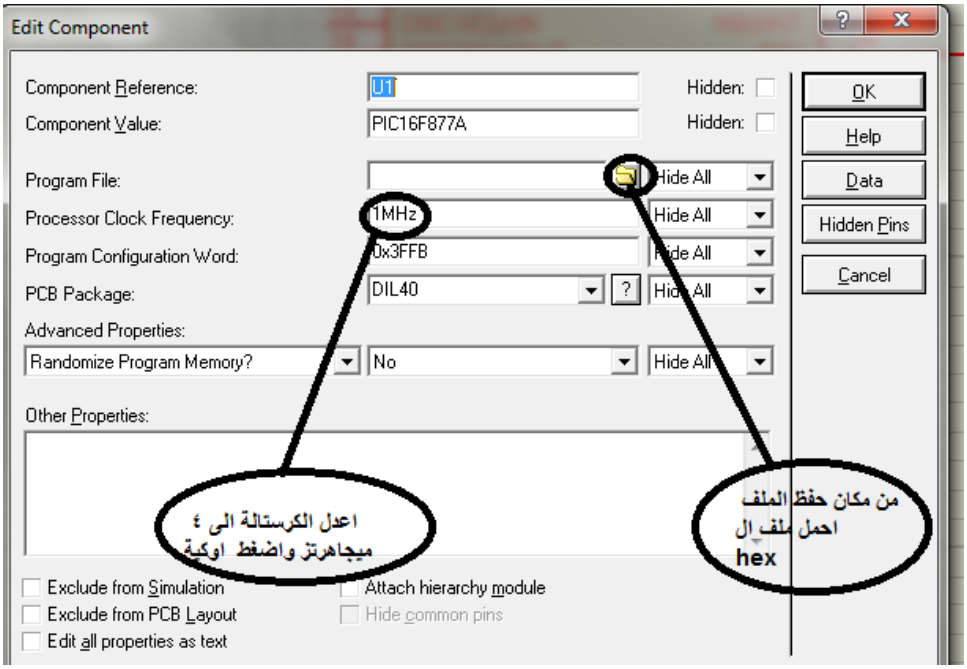
R= (5-1.2)/25mA=152Ω

إذن فإن أقل مقاومة يمكن استخدامها بحيث لا تؤثر على ال pin هي 152Ω ويفضل أن تكون قيمة المقاومة أكبر من ذلك لكي لا يتم استهلاك أقصى قيمة للتيار من ال pin

لذا الموجود فى السوق المقاومة220 اوم لذا اضعها قبل الليد



أقوم بتحميل ملف ال hex من نفس المكان اللى حفظت كود ال c فيه كما هو مبين بالشكل اضغط على الميكرو ضغطتين ستظهر هذة النافذة

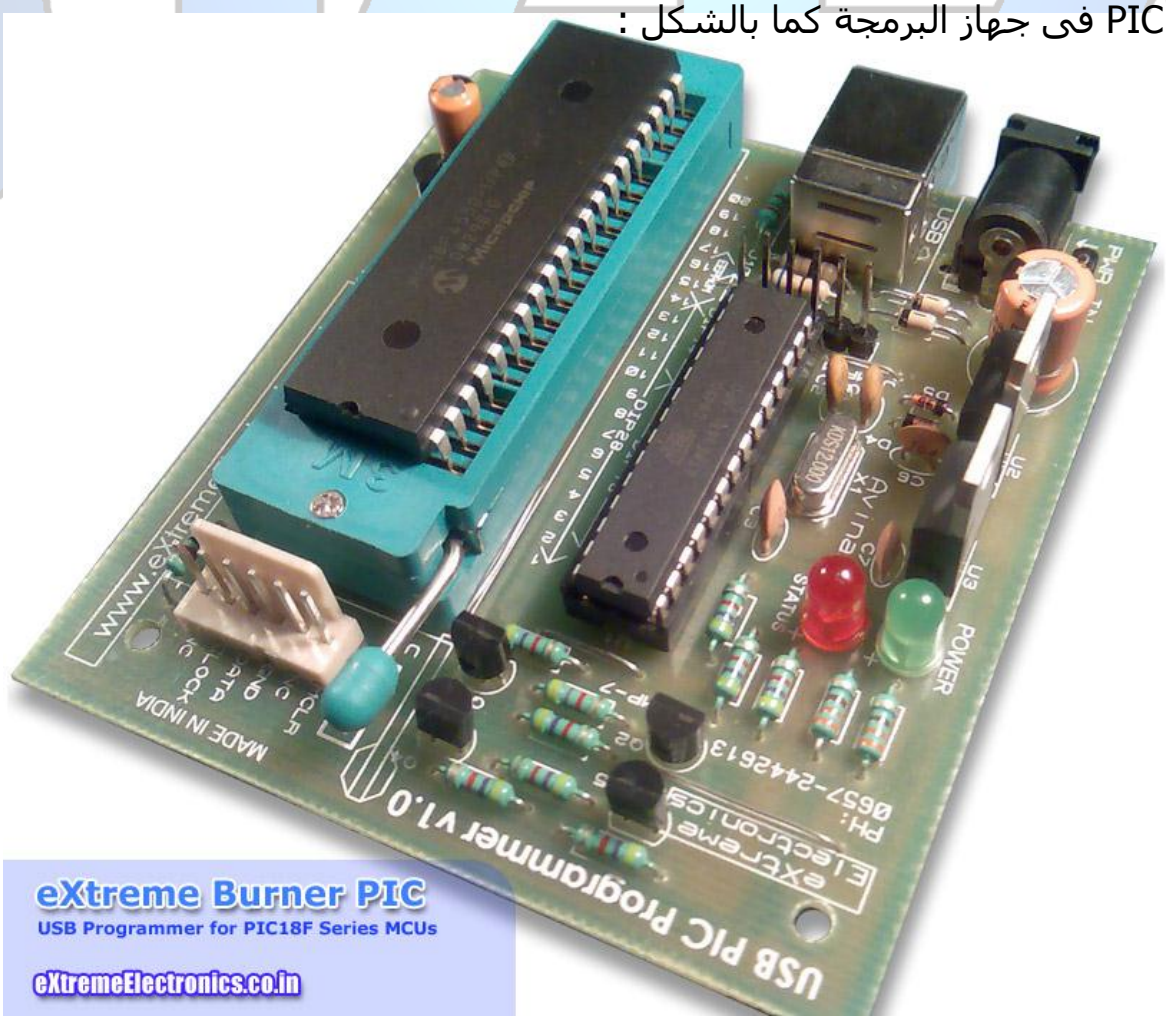


وعند عمل RUN للدائرة سأجد انها تضئ ثانية وتنطفئ ثانية .

Explain of winpic program

طريقة حرق البرنامج على ال PIC :

سنضع ال PIC فى جهاز البرمجة كما بالشكل :



يتم الحرق بواسطة الاستعانة ببعض برامج البرمجة مثل ال WINPIC وهو خاص لحرق البيك عن طريق توصيل البيك بالكومبيوتر بواسطة كابل ال SERIAL :

لحرق البرنامج على ال PIC

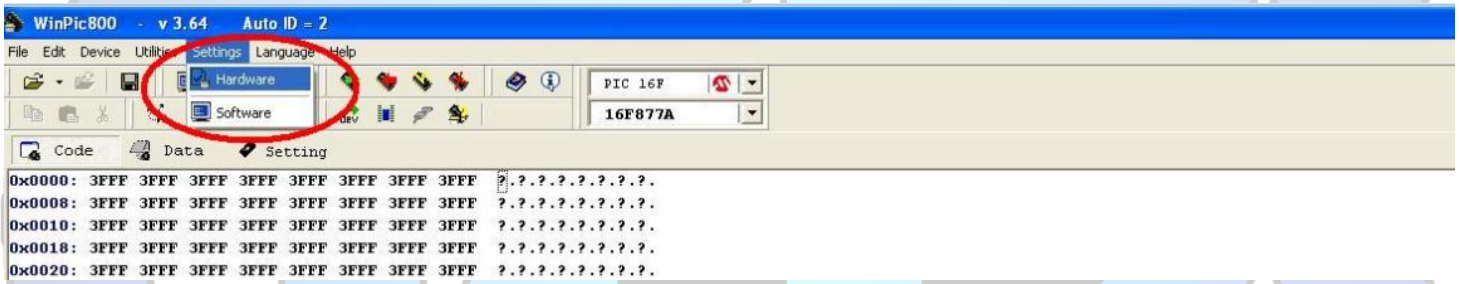
1_ اتاكد ان المفتاح اللى على ال kit لل programmer فى وضع ال burn mode .

2_ اضع الميكرو واشيله من ال socket وال power مقفولة .

3_ لحرق الميكرو باستخدم برنامج winpic800 .

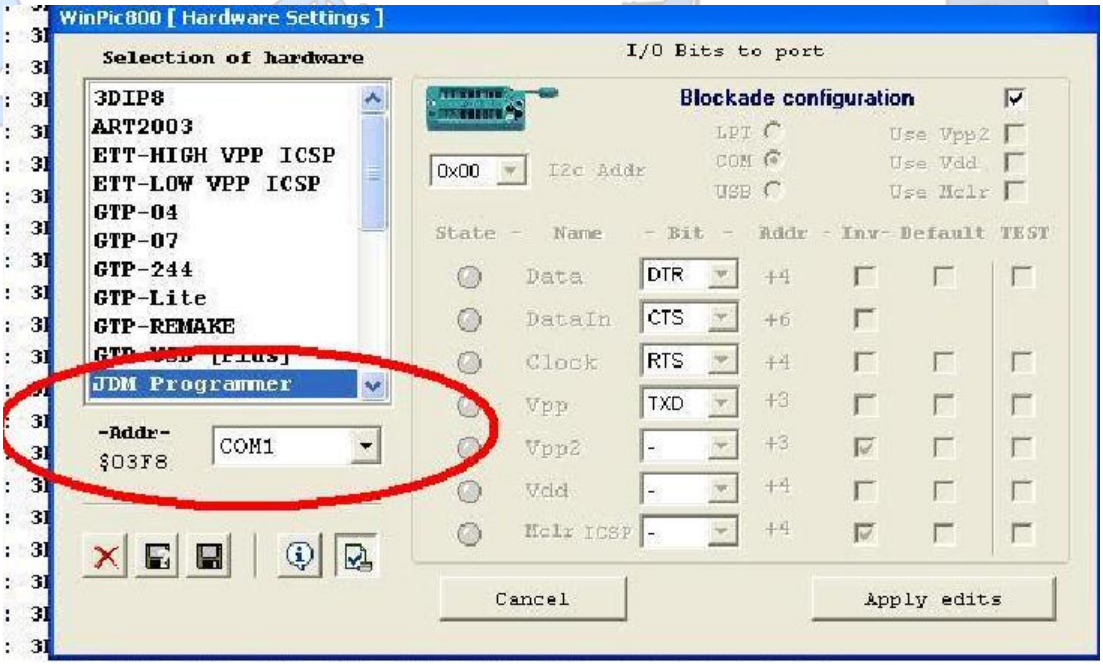
لازم الاول احدد نوع ال programmer من ع البرنامج لذا من قائمة settings اختار hardware

الجهاز المستخدم من النوع JDM لذا :

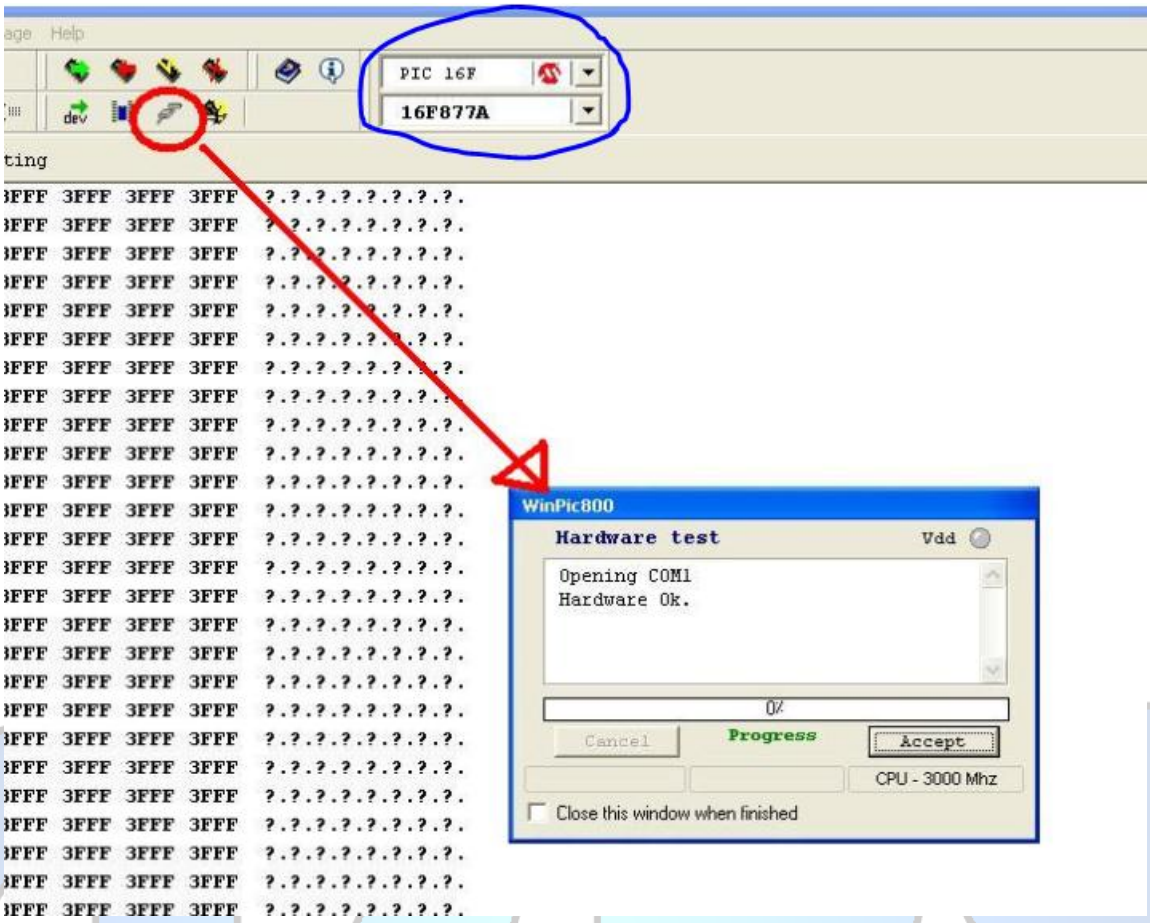


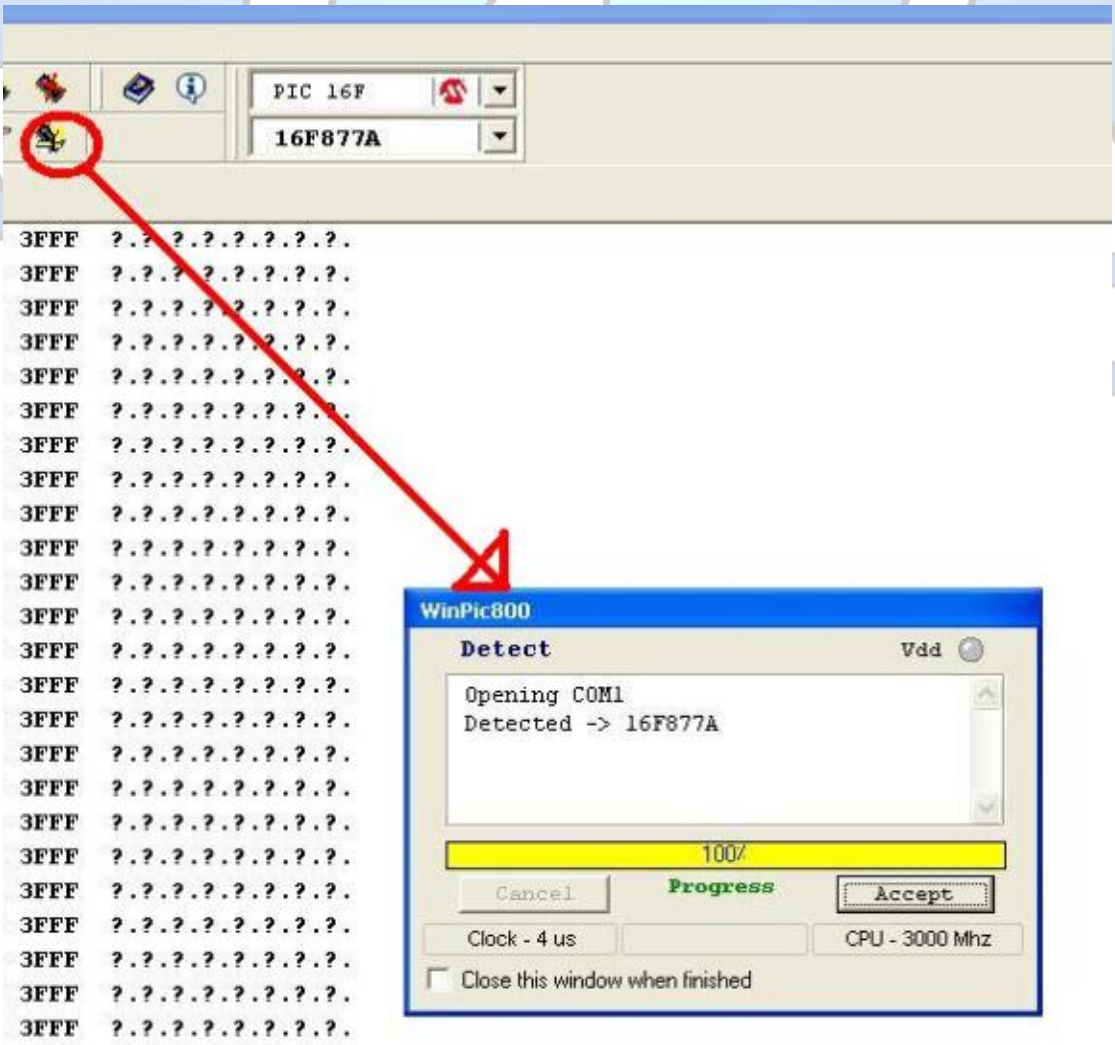
*اختار JDM programmer

*Com1 ثم اضغط apply



*اتاكد ان نوع ال pic 16 من ع الجنب الايمن اختار من قائمة device اختار hardware test


PIC 16F
16F877A
WinPic800
Hardware test
Vdd
Opening COM1
Hardware Ok.
0%
Cancel
Progress
Accept
CPU - 3000 Mhz
Close this window when finished

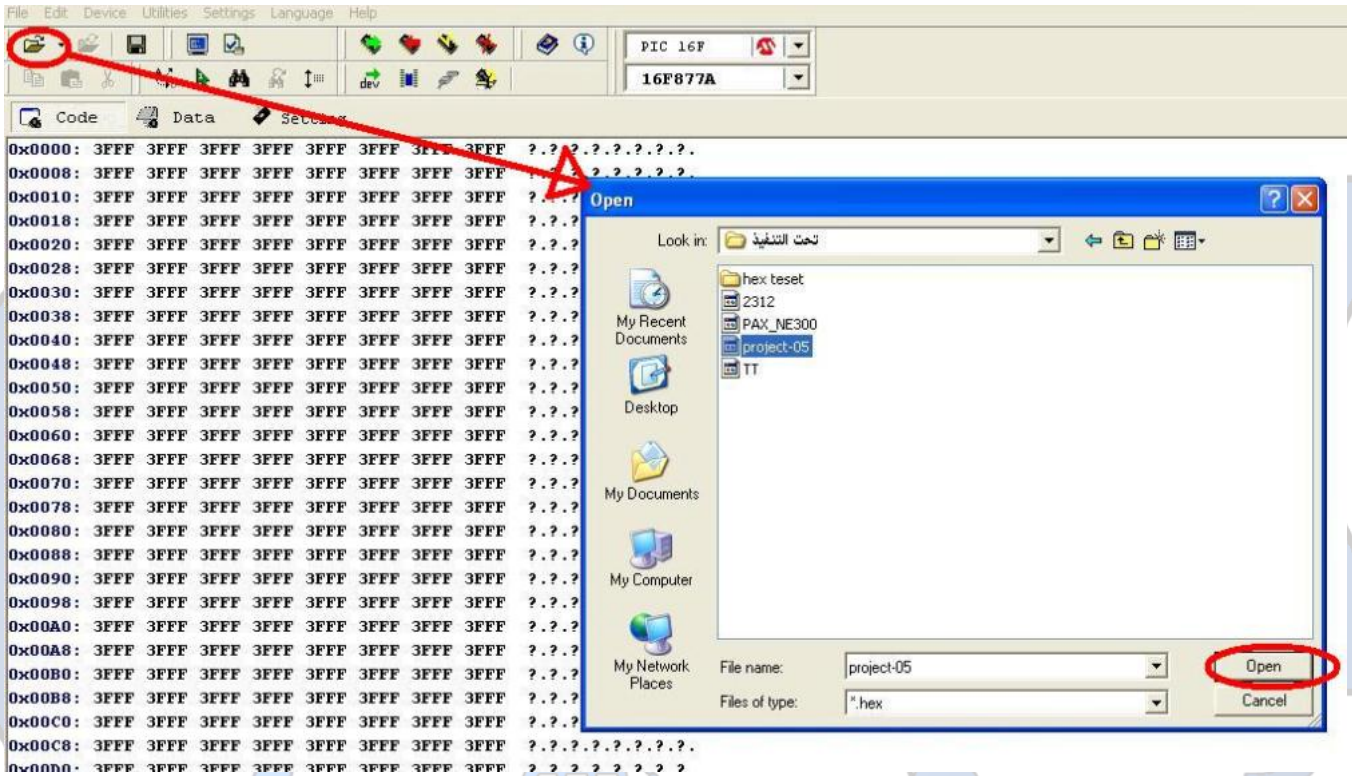

PIC 16F
16F877A
WinPic800
Detect
Vdd
Opening COM1
Detected -> 16F877A
100%
Cancel
Progress
Accept
Clock - 4 us
CPU - 3000 Mhz
Close this window when finished

* ثم من قائمة device اختار detect device عشان اتاكد ان الميكرو واصل لو مكتوب unknown معنى كده الميكرو مش واصل طب لو ادى unknown ممكن يكون احد الاحتمالات الاتية

_ مش ع ال burn mode اعمله run ثم ارجعه burn mode

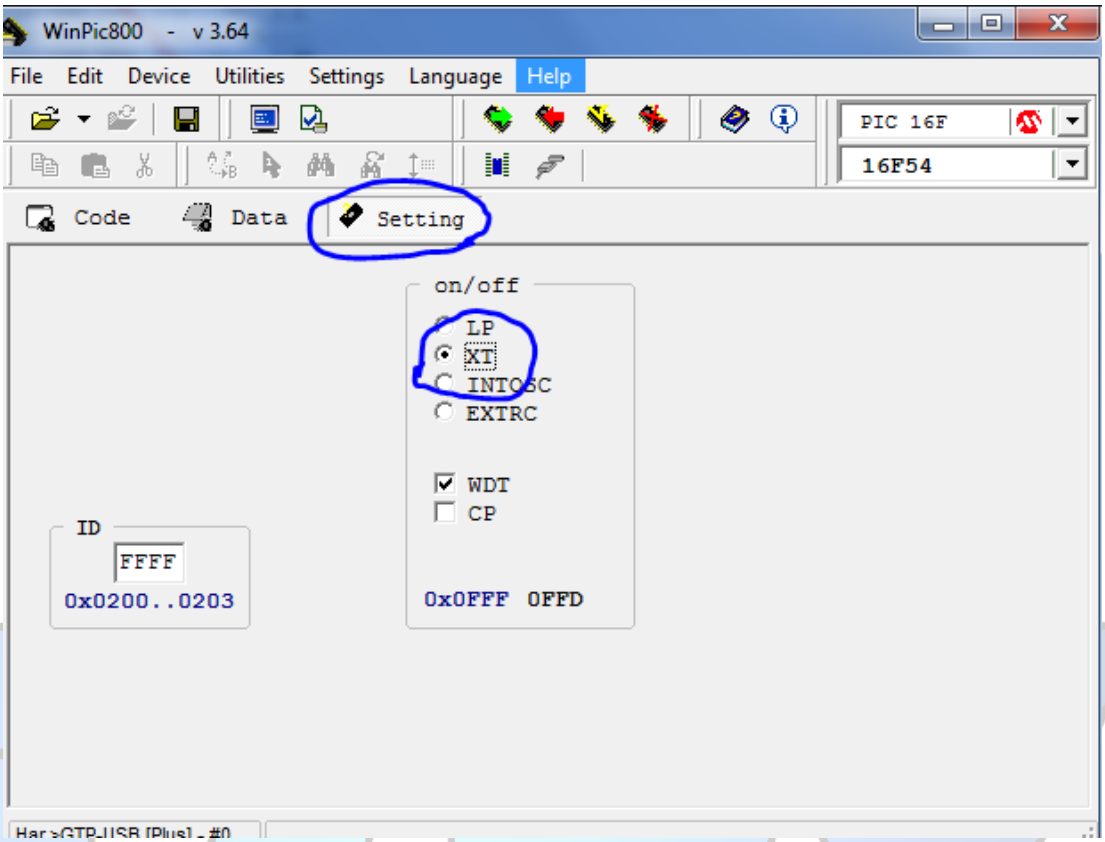
_ال serialمش داخل كويس

_ ممكن كابل ال serial بايظ او ال kit بايظه

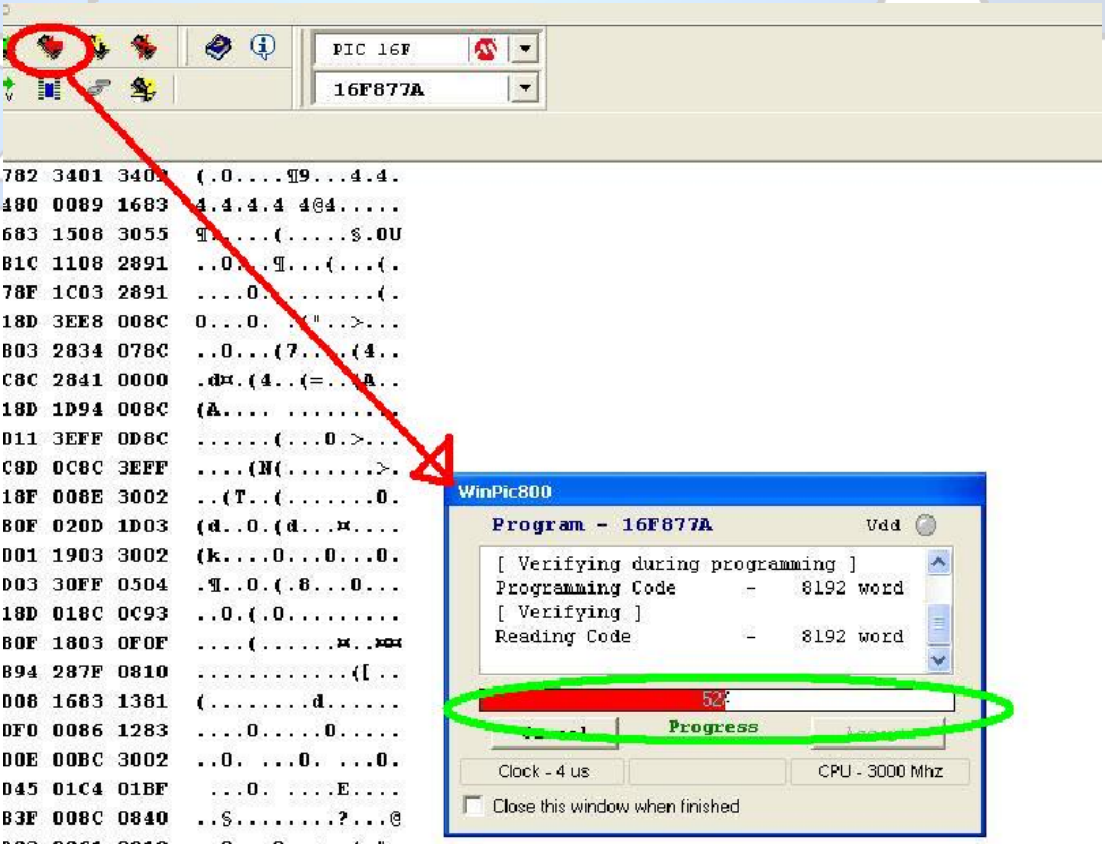


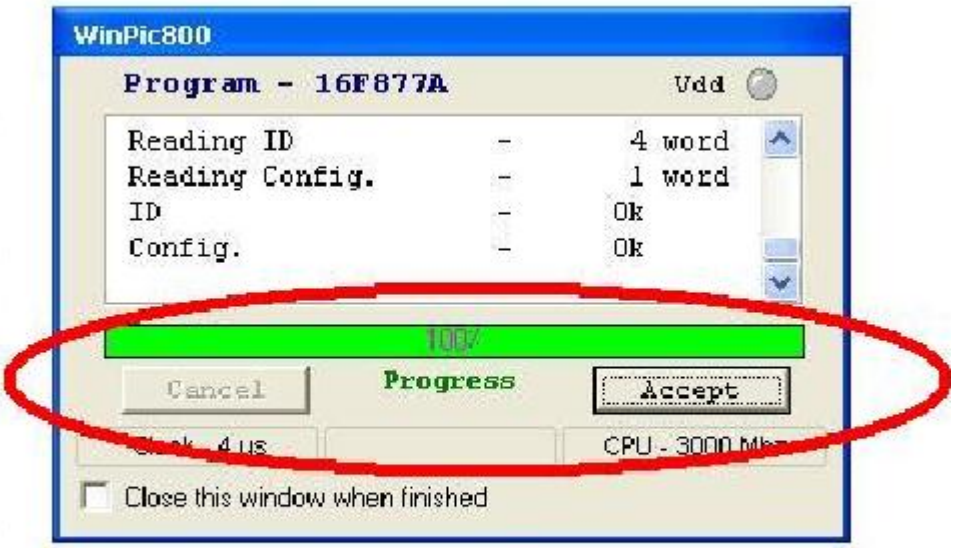
من قائمة file اختار open ثم اختار المشروع اللى عاوزة احرقه ع ال pic بامتداد hex.

من setting اختار من osc اختار منها XT وتاكد ان الاختياران WDTEN و LVP مش عليهم صح UNCHECKED



ثم اضغط PROGRAM ALL




WinPic800
Program - 16F877A
Vdd
Reading ID - 4 word
Reading Config. - 1 word
ID - Ok
Config. - Ok
100%
Cancel
Progress
Accept
CPU - 3000 Mhz
Close this window when finished

المشاريع

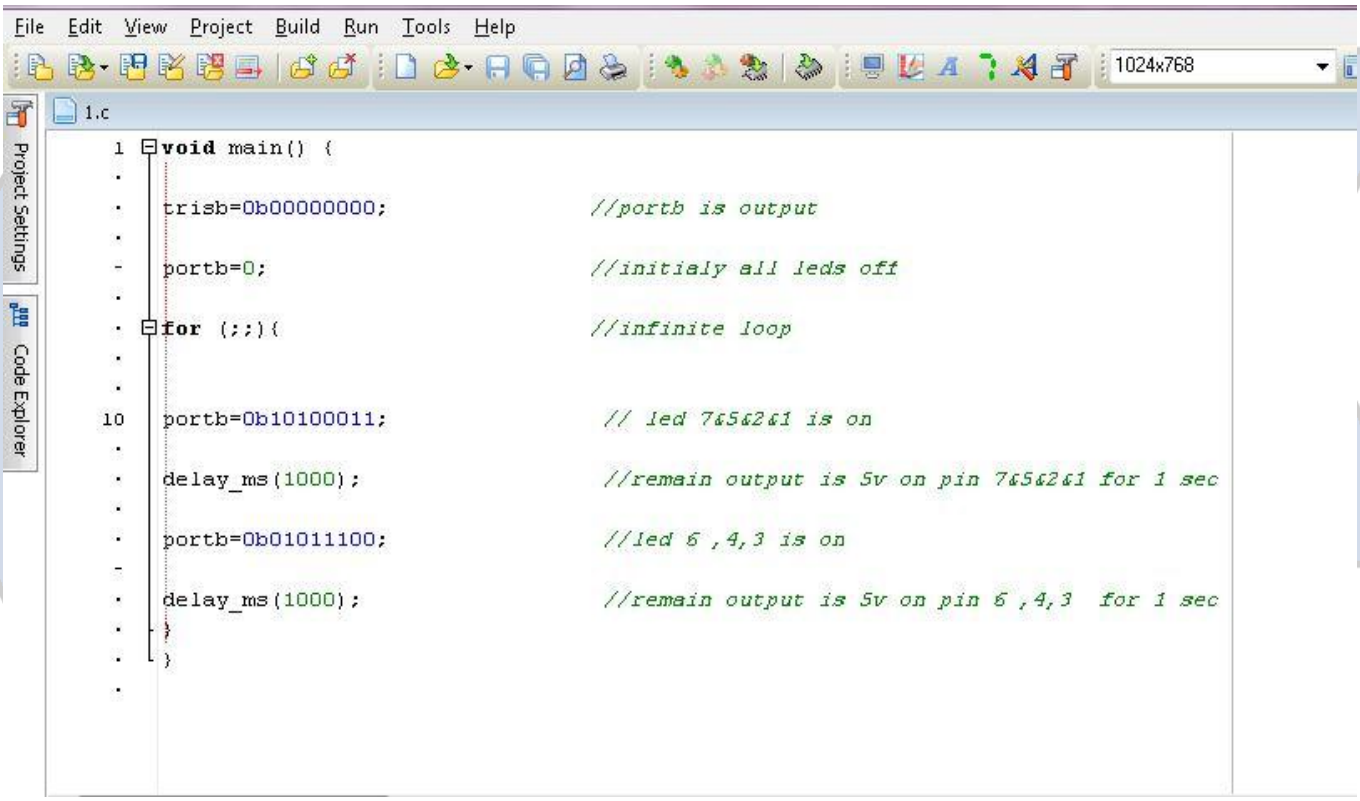
المشروع الاول

فكرة المشروع :

الهدف هو توصيل 8 ليدات على portb الليدات الموصلة بالرجل B0,B1,B5,B7 تضىء لمدة ثانيتين والباقى مطفى وبعد مرور ثانيتين تضىء B2,B3,B4,B6 والباقى ينطفئ وهكذا .

البرنامج :

نقوم بانشاء مشروع جديد وكتابة فيه هذا الكود




```
void main() {

trisb=0b00000000;              //portb is output

portb=0;                       //initialy all leds off

for (;;){                      //infinite loop


portb=0b10100011;               // led 7&5&2&1 is on

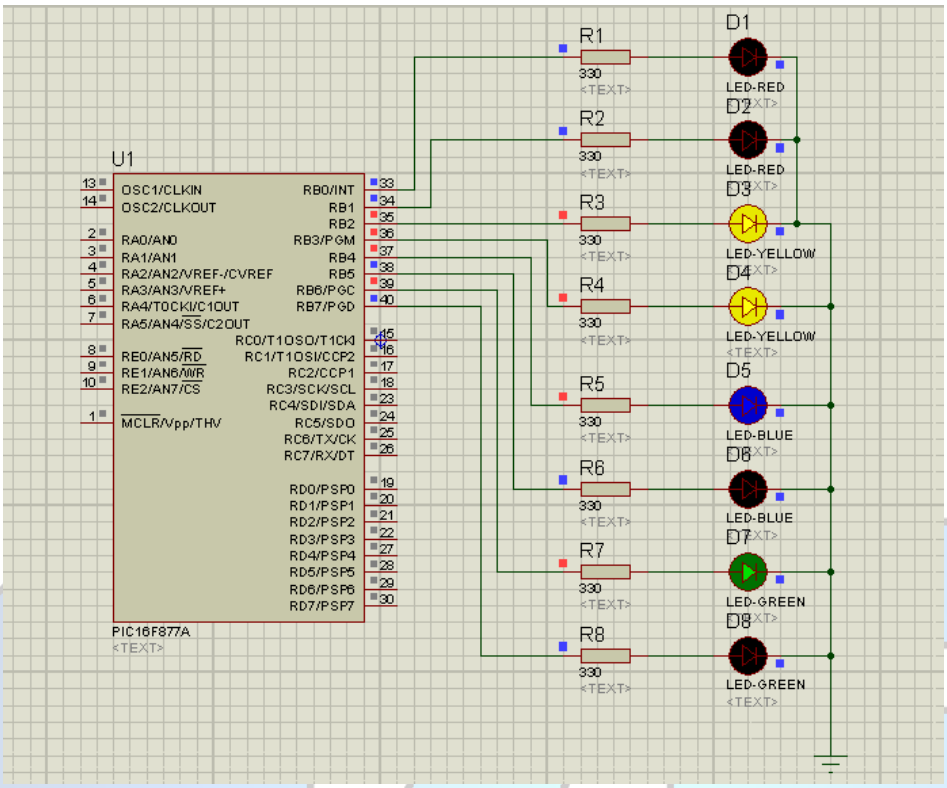
delay_ms(1000);                 //remain output is 5v on pin 7&5&2&1 for 1 sec

portb=0b01011100;               //led 6 ,4,3 is on

delay_ms(1000);                 //remain output is 5v on pin 6 ,4,3  for 1 sec
}
}
```

بالنسبة للسطر ;TRISB=0B00000000 بما أن كل البتات (Bits) قيمتها بأصفار نستطيع أن نكتب هذا السطر البرمجي بطريقة أخرى كالتالي ;TRISB=0 وسيؤدي نفس الوظيفة .

توصيل الدائرة :



ملحوظة :

من الممكن أن نضم الاوامر الى بعضها دون الحاجة الى أن نضغط enter لكن لا يستحب ذلك حتى يكون شكل البرنامج منظم ويسهل مراجعته وفهمه .

المشروع الثانى

فكرة المشروع :

انارة 8 ليدات لمدة ثانيتين وإطفاءهم ثانيتين .

البرنامج :

نقوم بانشاء مشروع جديد وكتابة فيه هذا الكود

2.c

```
void main() {

trisb=0b00000000;                //portb is output

portb=0;                         //initialy all leds off

for (;;){                        //infinite loop

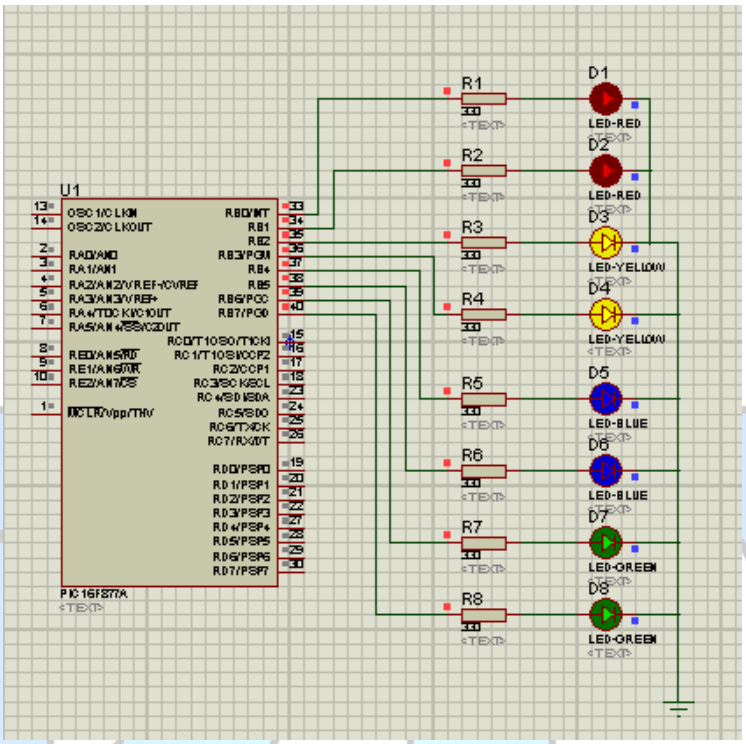

portb=~portb;                  // initially all leds off after 2 sec change state of leds to on

delay_ms(1000);                   //remain output is 5v on pin 7&5&2&1 for 1 sec


}
}
```

ابتدائا كانت الليدات مطفية ثم جاء الامر بعكس حالة الليدات فتضىء ثم تعكس حالتها فتطفى وهكذا .

توصيل الدائرة :



المشروع الثالث

فكرة المشروع :

برنامج flasher لكن الاربع ليدات الاوليين ينوروا ثم ينطفؤا وينور الاربعه اللى بعدهم .

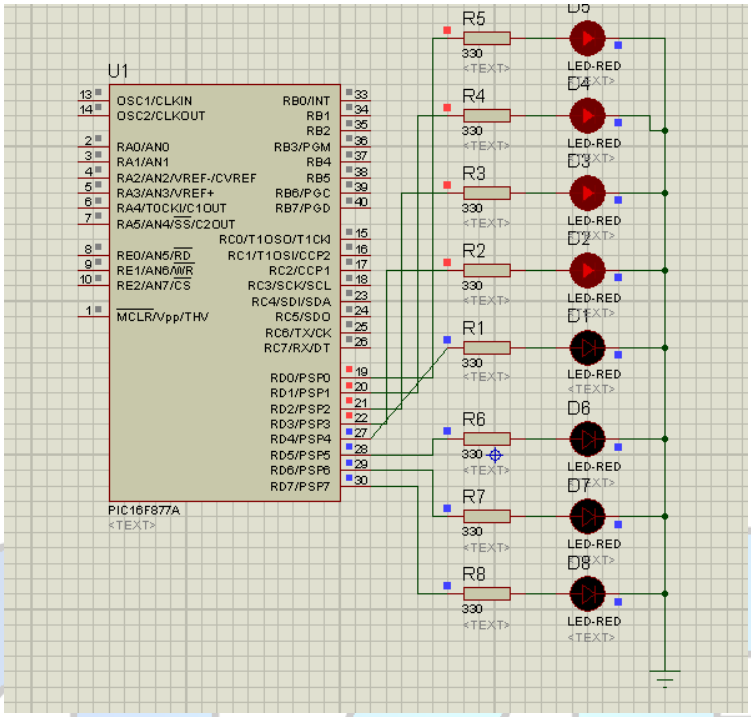
البرنامج :

نقوم بانشاء مشروع جديد وكتابة فيه هذا الكود

3.c

```
void main() {
TRISD=0;
portd=0;
for(;;){
                PORTD=0b00001111;
        delay_ms(1000);
                PORTD=0b11110000;
        delay_ms(1000);
}
}
```

توصيل الدائرة :



المشروع الرابع

فكرة المشروع :

تشغيل ال led الموصل على b0 عند الضغط على المفتاح واطفاءة عند عدم الضغط .

البرنامج :

نقوم بانشاء مشروع جديد وكتابة فيه هذا الكود .

فى هذا المشروع نعرف المفتاح على انه دخل المقتاح موصل ACTIVE LOW بمعنى انه اذا كان مضغوط فيصل ل pin الميكرو 0v وان لم يكن مضغوط سيصل 5v .

4.c

```
void main() {

 trisc.f0=0;                    //RC0 IS OUTPUT

 trisb.f0=1;                    //RB0 IS INPUT

 portc.f0=0;                    //LED IS OFF INITIALLY

 for(;;){                       //INFINITE LOOP

 if (portb.f0==0){              // IF SWITCH IS PRESSED

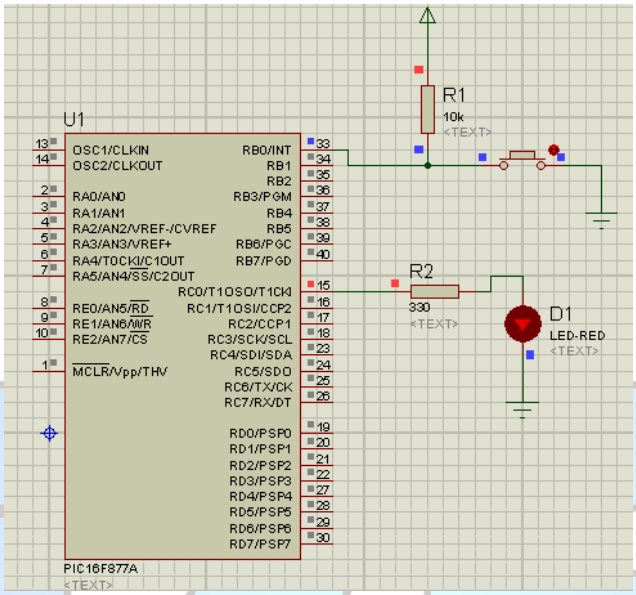
     portc.f0=1;                //LED IS ON

     }
     if (portb.f0==1){          //IF SEITCH ISNOT PRESSED

     portc.f0=0;                //LED IS OFF
     }
     }
}
```

ولهذا تم استخدام الامر if واستخدم == للمقارنة فاذا كان الدخل = zero معنى كده ان المفتاح مضغوط فاطلع خرج 5 فولت على الليد واذا كان المفتاح غير مضغوط تنطفىء الليد .

توصيل الدائرة :



المشروع الخامس

فكرة المشروع :

دا برنامج لما اضغط على ال push button الاول تنور الليد الاولى فقط لما اضغط على ال push button التانى تنور الليد التانية فقط لما اضغط على ال push button الثالث تنور الليد الاولى والثانية معا .

البرنامج :

نقوم بانشاء مشروع جديد وكتابة فيه هذا الكود .

```
// buttons are connected to RB0, RB1,AND RB2
//LEDS ARE CONNECTED TO RC0 AND RC1
void main (){
TRISB=255;                                  //PORTB INPUT

TRISC=0;                                    //PORTC OUTPUT

PORTC=0;                                    //initially leds off

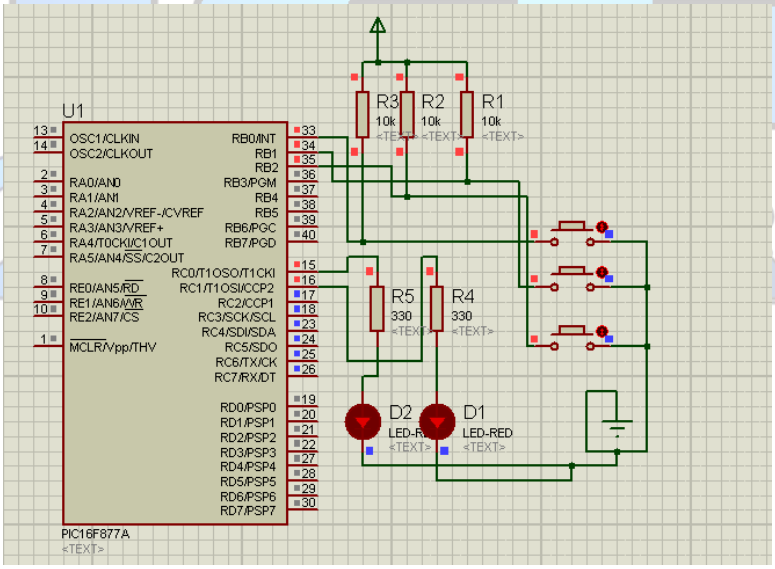
for (;;){                                   //infinite loop

        if (RB0_bit==0){                  //IF BUTTON 1 IS PRESSED

RC0_bit=1;                                          //LED 1 ON

RC1_bit=0;                                        //LED 2 OFF
 }
        else if (RB1_bit==0){         //IF BUTTON 2 IS PRESSED
RC1_bit=1;                                          //LED 2 ON
RC0_bit=0;                    //LED 1 OFF
}
else if (RB2_bit==0){                 //IF BUTTON 3 IS PRESSED
        RC0_bit=RC1_bit=1;                          // BOTH LEDS ON
}
}
}
```

توصيل الدائرة :



المشروع السادس

فكرة المشروع :

عاوز اعمل برنامج بينور ليد واحدة اول ما اضغط على ال Push button ويطفى الليد اول ما اضغط ع ال push button تانى ؟؟؟؟

البرنامج :

وانا بفكر فى البرنامج اولا هخلى PORTB هو الدخل و C هو الخرج والحالة الابتدائية لليد مطفئة لما اضغطت ع ال push button يغير حالة الليد لو منورة تطفى والعكس الكلام ده ممكن اترجمه فى صورة الكود ده

```
void main (){
TRISB=255;
TRISC=0;
rc0_bit=0;
for (;;){
if (rb0_bit==0){
rc0_bit=!rc0_bit;
}
}
}
```

بس فى مشكله دلوقت ان الميكرو بينفذ الامر فى u sec يعنى الميكرو هينفذ الامر قبل ما الشخص يشيل ايده من ع ال push button فمحتاج ان الميكرو يتاخر شوية طب الحل ممكن يكون انى اعمل delay بس الفكرة دى مش كويسة لانى معرفش ال user هيشيل ايده امتى الحل التانى انى اعمل loop تعمل check ال user شال ايده ولا لا اضيف ع الكود While (RB0==0){} يعنى طول ما المستخدم ضاغط ع ال push button افضل ادور فى loop من غير ما تعمل حاجه اول ما يشيل ايده اعكس حالة الليد .

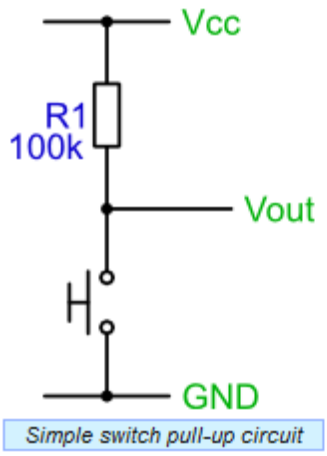
سيصبح الكود هكذا :

```
void main (){
TRISB=255;
TRISC=0;
rc0_bit=0;
for (;;){
if (rb0_bit==0){
rc0_bit=!rc0_bit;
 while (RB0_bit==0){}
}
}
}
```

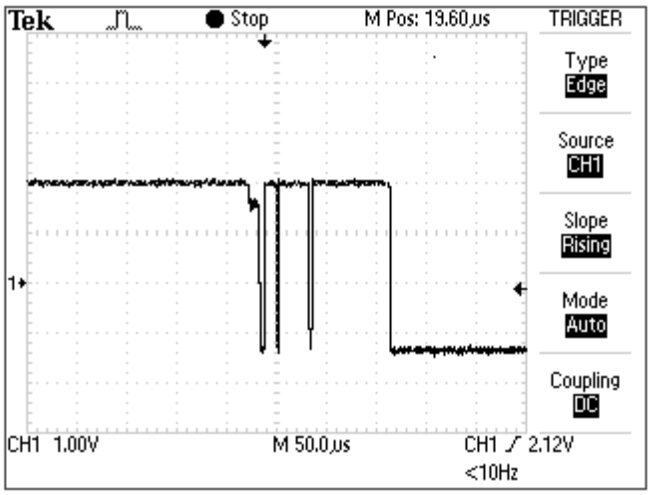
فى مشكلة تانية هتواجهنا ان اى mechanical button لما اضغط عليه فان اجزاءه بتنضغط بيفضل يرتعش شوية لحد ما يستقر تانى وبيعمل bounce الميكرو بيترجم ال bounce على انها اكتر من press الزمن لل bounce مش بيزيد عن 10m sec انا دلوقت عارف الزمن ممكن اعمل delay وعمر ما المستخدم هيلحق يشيل ايده فى الزمن الصغير ده ال button بيعمل bounce وهو بينضغط وهو بيطلع عشان كده اضع delay مرتين .

لاحظ لو شغلت البرنامج من غير delay على البروتس هيشتغل ويكون كويس لان البروتس مش بيعمل simulate لل mechanical bouncing .

فى مشكلة ايضا لو وضعت الدايرة فى noisy environment الميكرو بيفسر ال noise على انها press لازم افرق ال noise عن ال key press طيب ال noise بتقعد فترة قليلة اقل من 10m sec يعنى اشوف بعد مرور 10m sec لو لو rb0==0 اقلب حالة الليد يعنى انا لو عدت noise من غير ما اضغطت والميكرو فسرها على انها press بعد 10 ملى ثانية هيشوف لو لقى high يعنى دى noise هيتجاهل الامر لو لقاها low هيعرف انها press وينفذ الامر ويعكس حالة الليد



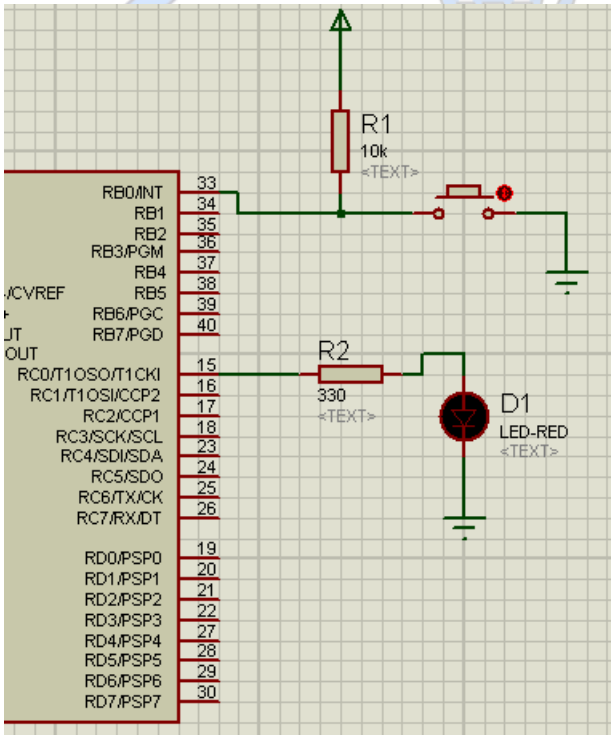
Simple switch pull-up circuit



Switch bounce produced on switch press

الكود النهائى بعد التعديل

```
void main (){
TRISB=255;
TRISC=0;
rc0_bit=0;
for (;;){
if (rb0_bit==0){
     delay_ms (10);
         if (rb0_bit==0){
rc0_bit=!rc0_bit;
 while (RB0_bit==0){}
  delay_ms (10);
}
}
}
}
```



توصيل الدائرة :

المشروع السابع

فكرة المشروع :

برنامج PIR عندما يدخل شخص الغرفة ال PIR يحس ويعطى HIGH فاللمبة تنور لما الشخص يخرج اللمبة تطفى لو اراد الشخص الجلوس فى الظلمة يضغط على SWITCH يطفى النور ولو خرج من الغرفة ودخل النور يكون مطفى لان اخر حالة لل SWITCH انه كان مطفى اما لو كان خارج من الغرفة والنور مفتوح لما يدخل الغرفة تانى هيكون النور مفتوح

البرنامج :

عاوز انا register يحتفظ بأخر حالة ليه لذا هنعرفة على انه x=0 unsigned char حالتة الابتدائية صفر عندى 2 دخل ال PIR sensor وال switch وعندى خرج واحد وهو اللمبة .

```
void main (){

unsigned char x=0;      //declare variable x

TRISB=0b11111011;       //rb2 is output and other pins are input

RB2_bit=0;               // الحالة الابتدائية للمبة انها مطفية

for (;;){                //infinite loop

if (RB1_bit==0){                    // BUTTON IS PRESSED
delay_ms (10);
if (RB1_bit==0){
x=!x;

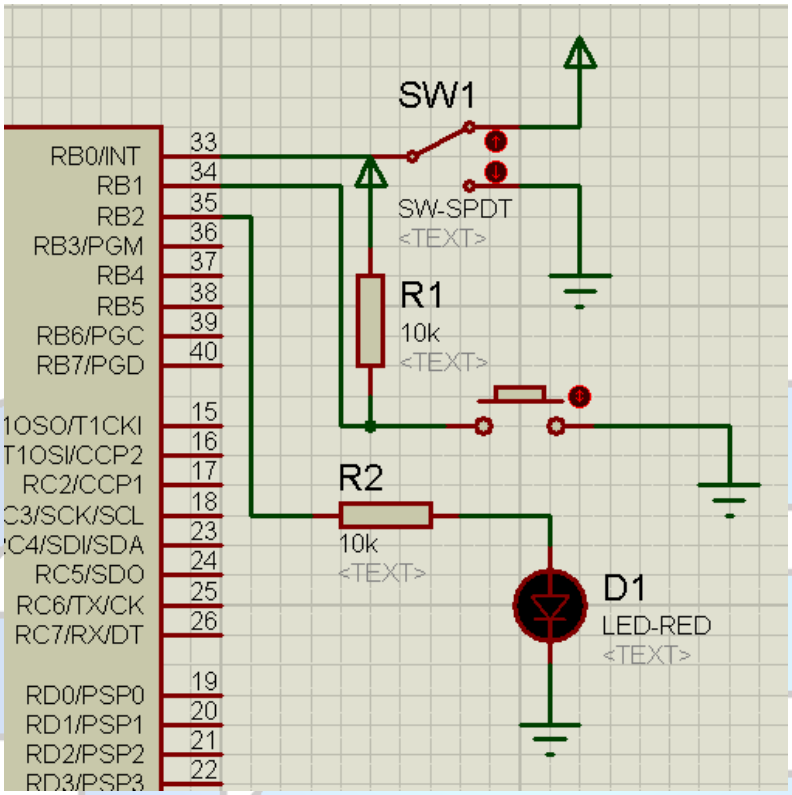
while (RB1_bit==0){}
delay_ms (10);
}
}
if ( (RB0_bit==1)&&(x==0)){       // button not pressed & someone enter the room
RB2_bit=1;                 //LAMP IS ON
}
else {
RB2_bit=0;
}
}
}
```

نشوف هل البرنامج سليم ولا لا لو الشخص دخل يعنى هيطبق حالة IF الثالثة لان الشرطين متوفرين الشخص دخل الغرفة و x=0 طبقا لقيمتها الابتدائية فاللمبة هتنور لو خرج هيطبق حالة else واللمبة تطفى لكن لو دخل اللمبه هتنور ولو داس ع المفتاح فان x هتعكس حالتها وتصبح واحد وقتها شرط if مش هيتحقق لان x=1 وبالتالى اللمبة هتنطفى لو ضغط على ال switch مرة تانيه x هتعكس حالتها وتصبح zero وبالتالى شرط if هيتحقق الشخص بالغرفة و x=0 فاللمبة هتنور تانى طب لو خرج من الغرفة اللمبة هتطفى لو دخلها تانى اللمبه هتنور

زى اخر حالة هو سايبها عليها طب لو ضغط على ال switch فان x هتعكس حالتها وتصبح x=1 فاللمبة هتطفى لو خرج من الغرفة وبعد كده دخلها تانى هتكون x=1 والشخص فى الغرفة فاحد الشرطين مش متوفر فاللمبة مش هتنور وبكده حققت البرنامج اللى انا محتاجه .

توصيل الدائرة :

عندما يطبق هذا البرنامج على بروتس ممكن نعمل simulate لل PIR ب switchمن النوع SW-SPDT يعطى مرة VCC اذا كان الشخص دخل ومرة GROUND اذا خرج الشخص .



المشروع الثامن

فكرة المشروع :

عداد ثنائى binary أوتوماتيكى باستخدام الليدات (8 ليدات فقط)

العداد المكون من 8 ليدات يستطيع العد من 0_255 وذلك لان 11111111تكافئ 255 بال decimal .

البرنامج :

نقوم بانشاء مشروع جديد وكتابة فيه هذا الكود .

العد بال binary يكون كالاتى :

```
OOOOOOOO
OOOOOOO1
OOOOOO1O
OOOOOO11
OOOOO1OO
.........
.........
11111111
```

الامر portb=0 يكافئ portb=0b00000000; وايضا portb=1; يكافئ portb=0b00000001; وهكذا .

كود البرنامج :

```
void main(){
TRISB=0;                    //portb output

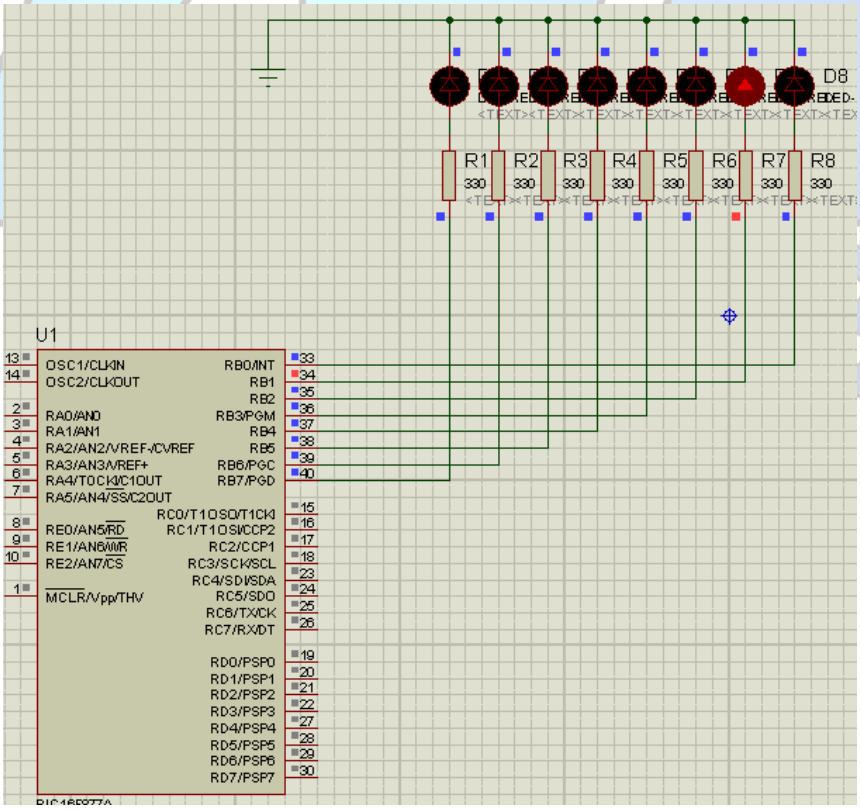
PORTB=0;                    //leds off initially
delay_ms(1000);             //delay to see zero count

for(;;){     //infinite loop

PORTB++;     //increment by one

delay_ms(1000);          // delay 1 sec
                }
}
```

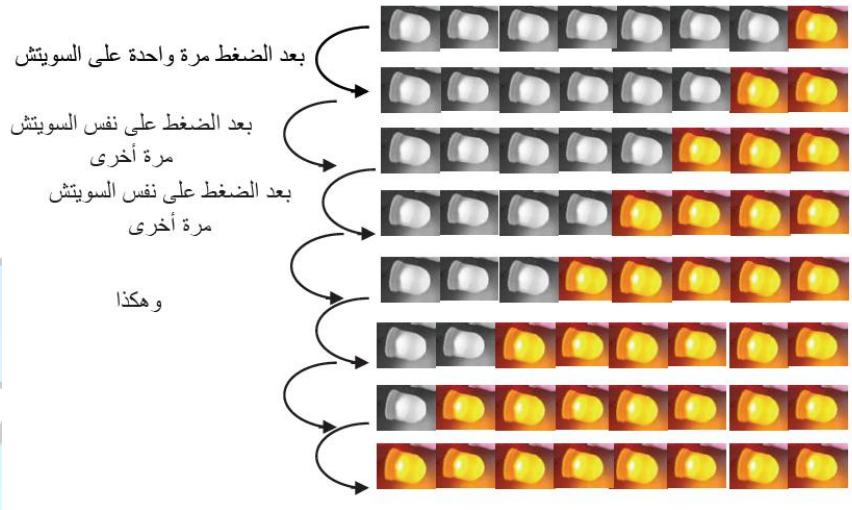
توصيل الدائرة :



المشروع التاسع

فكرة المشروع :

عند الضغط على ال switch الموصل ب C0 يضئ ليد معين B0 ثم اذا ضغطنا مرة أخرى على نفس ال switch يضئ هذا ال led والذى يليه وهكذا الى ان تضئ ال 8 ليدات فاذا ضغطنا بعد ذلك على السوتش تطفئ جميعا



طريقة التفكير فى البرنامج :

طبيعى هنستخدم جملة if عشان تختبر هل ضغطت على ال switch أم لا لكى يضئ الليد B0 لكن كيف سيعرف أنى ضغطت على ال switch مرة أخرى لكى يضئ الليد التى تليها , من الممكن التفكير فى وضع جملة if ثانيه بداخل الاولى تختبر هل تم الضغط على ال switch مرة ثانية أم لا وبالتالى يمكن كتابة الكود هكذا:

```
if (portc.f0=0){
portb.f0=1;
delay_ms (1000);
if (portc.f0=0){
potb.f1=1;
}
}
```

فبالتالى البيك سيختبر هل ضغط على المفتاح أم لا فاذا ضغط هيضئ اليد B0 ثم ينتظر ثانية ويختبر هل تم الضغط على ال SWITCH مرة ثانية أم لا فاذا تم الضغط عليه مرة ثانية يضئ الليد B1 ولكن اذا لم يتم الضغط ينتقل الى الامر الذى بعده ثم الى الذى بعدة وعندما اقوم أنا بالضغط على ال SWITCH سيضئ ليد غير اللى انا محتاجها ... طب ايه الحل ؟؟؟؟

الحل انى اعمل loop يفضل يلف فيها لحد ما اضغط على ال switch مرة ثانية

While (rc0_bit==1){};

يفضل يلف فى ال loop طول ما ال switch مش مضغوط

ويصبح الكود بهذا الشكل :

```
void main() {
trisb=0;        //portb is output

trisc=255;      //portc is input

portb=0;        //all leds off initially

for (;;){       //infinite loop

 if (rc0_bit==0){         //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f0=1;              //ler 1 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){         //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f1=1;               //ler 2 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f2=1;                //ler 3 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f3=1;                //ler 4 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
```

```
delay_ms (10);
if (rc0_bit==0){
portb.f4=1;                 //ler 5 is on
while (rc0_bit==0){};
delay_ms (10);
 while (rc0_bit==1){};
}
}
if (rc0_bit==0){            //if switch is pressed
delay_ms (10);
if (rc0_bit==0){
portb.f5=1;                   //ler 6 is on
while (rc0_bit==0){};
delay_ms (10);
 while (rc0_bit==1){};
}
}
if (rc0_bit==0){            //if switch is pressed
delay_ms (10);
if (rc0_bit==0){
portb.f6=1;                  //ler 7 is on
while (rc0_bit==0){};
delay_ms (10);
 while (rc0_bit==1){};
}
}
if (rc0_bit==0){            //if switch is pressed
delay_ms (10);
if (rc0_bit==0){
portb.f7=1;                   //ler 8 is on
while (rc0_bit==0){};
delay_ms (10);
 while (rc0_bit==1){};
}
}
if (rc0_bit==0){            //if switch is pressed
delay_ms (10);
if (rc0_bit==0){
portb=0;                      //lers is off
while (rc0_bit==0){};
delay_ms (10);
 while (rc0_bit==1){};
}
}
}
}
```

لكن فى مشكلة ان البيك هتفضل ماشية فى البرنامج هتعدى if الاولى والتانية وهكذا لحد ما اضغط ع ال switch ويبدأ ينفذ البرنامج فممكن يبدأ يشتغل من عند الليد الثالثة طب أنا عاوز اول ليد هى اللى تنور أعمل ايه ????

اضع شرط ال loop فى اول البرنامج بحيث يفضل يلف فى لوب لحد ما اضغط ع المفتاح ويبدأ ينور ال led الاولى .

ليصبح الكود النهائى

```
void main() {
trisb=0;        //portb is output

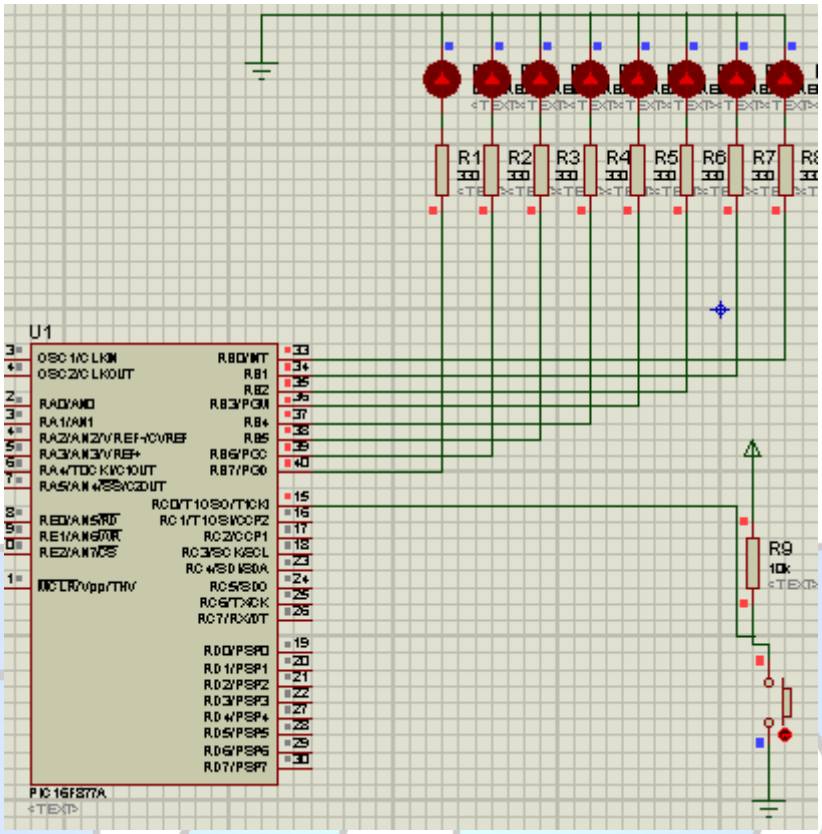
trisc=255;      //portc is input

portb=0;        //all leds off initially

for (;;){       //infinite loop
      while (rc0_bit==1){};
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f0=1;               //ler 1 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f1=1;                //ler 2 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){           //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f2=1;                 //ler 3 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){           //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f3=1;                 //ler 4 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
```

```
 delay_ms (10);
 if (rc0_bit==0){
 portb.f4=1;                //ler 5 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f5=1;                 //ler 6 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f6=1;                 //ler 7 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb.f7=1;                 //ler 8 is on
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
 if (rc0_bit==0){          //if switch is pressed
 delay_ms (10);
 if (rc0_bit==0){
 portb=0;                    //lers is off
 while (rc0_bit==0){};
 delay_ms (10);
  while (rc0_bit==1){};
}
}
}
}
```

توصيل الدائرة :



المشروع العاشر

سنقوم بعمل برنامج يقوم بنفس وظيفة البرنامج السابق لكن بطريقة اخرى .

الكود :

```
void main() {
 char x=0;       //declare variable x and initially is zero
 trisc.f0=1;  //rc0 is input
 trisb=0;      //portb is output
 portb=0;       //leds off initially
while (1){    //infinite loop
if (portc.f0==0){         //switch is pressed
delay_ms (10);
 if (portc.f0==0){
x++ ;
while (portc.f0==0){};
}
}
if (x==0) portb=0;
if (x==1) portb=0b00000001;
if (x==2) portb=0b00000011;
if (x==3) portb=0b00000111;
if (x==4) portb=0b00001111;
if (x==5) portb=0b00011111;
if (x==6) portb=0b00111111;
if (x==7) portb=0b01111111;
if (x==8) portb=0b11111111;
if (x==9) x=0;
}
}
```

خطوات سير البرنامج :

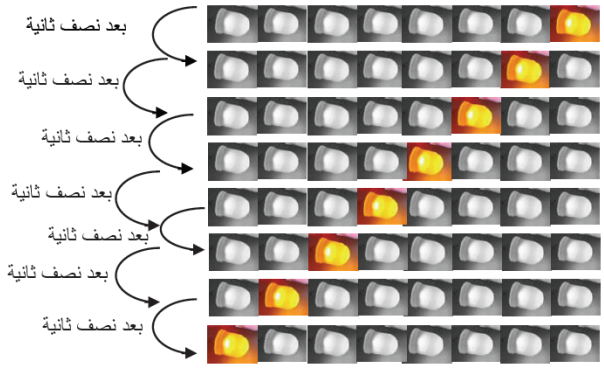
نقوم بالاعلان عن المتغير x واعطائة قيمة ابتدائية zero وعند كل ضغطه على ال switch يزيد قيمة المتغير x بمقدار واحد وكل قيمة للx تقوم بتنفيذ أمر معين فلنتابع ماذا يفعل البرنامج عندما يضغط المستخدم على ال switch تصبح x=1 وبالتالى ينفذ جملة if الثانية ويضئ الليد b0 وعند الضغط مرة ثانيه على ال switch تزداد قيمة x وتصبح 2 وينفذ جملة if الثالثة ويضئ الليد b0, b1 وهكذا الى أن تصبح x=8 وتضئ الليدات جميعا وعند الضغط على ال switch مرة أخرى تصبح x=9 فيقوم بجعل x=0 وبالتالى ينفذ جملة if الاولى ويطفئ الليدات جميعا وهكذا .

المشروع الحادى عشر

فكرة المشروع :

الهدف من هذا المشروع أولا هو التعرف على بعض الطرق البرمجية والتى ستسهل عليك المشاريع فيما بعد .

المطلوب إضاءة الليدات بالشكل التالى :



بالطبع هذا البرنامج سهل ويمكن إنجازه بعدة طرق من أسهلها أن نجعل portb=0b00000001; ثم بعد ذلك نجعل البك ينتظر نصف ثانية من خلال الأمر delay_ms(500); ثم بعد ذلك نجعل portb=0b00000010; وهكذا . نحن الآن سنقوم بعمل هذه التجربة بطريقة أخرى وبأمر برمجي جديد فلعلك تلاحظ أن الليد في كل خطوة يتحرك مرة لليسار (تتم إزاحته لليسار) فهل هناك أمر برمجي يقوم بعمل إزاحة لليسار ؟؟ بالطبع نعم وهو أمر بسيط جداً . انظر للكود التالي فسيقوم بالوظيفة المطلوبة من خلال استخدام هذا الأمر .

ملحوظة هامه :

يجب الاعلان عن المتغيرات أولا أى شئ بعد دالة main علطول

الكود :

```
void main() {
    char x;
trisb=0;        // portb is output

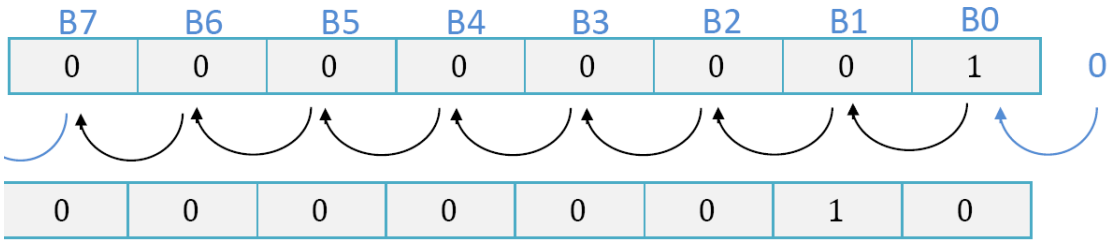
for (;;){
portb=0b00000001;
delay_ms (500);
for (x=0;x<8;x++){
portb=portb<<1;
      delay_ms (500);
}
}
}
```

تحليل الكود :

الامر portb=0b00000001; سيجعل الليد bo مضئ وننتظر نصف ثانية والهدف من جملة for هو تكرار الاوامر 8 مرات بدل من كتابة أمر الانتظار وأمر ال shift 8 مرات

أما الامر الهام وهو portb=portb<<1; سيقوم بعمل ازاحة لل bit ناحية اليسارفيجعل قيمة portb يساوى القيمة السابقة لل port مزاحة مرة الى اليسار فمن المعلوم أن المسجل portb يتكون من 8 bit وتتم الازاجه بهذا الشكل

شكل portb قبل عملية الإزاحة لليسار



شكل portb بعد عملية الإزاحة لليسار

فنرى ان ال bit رقم 1 سيحل مكانه ال bit رقم زيرو ويحل مكان ال bit رقم زيرو صفر من الخارج وهكذا ليصبح الخرج مع كل shift كالاتى :

00000001

00000010

00000100

00001000

00010000

00100000

01000000

10000000

فالعلامة >> تعبر عن ازاحة الى اليسار والعلامة << تعبر عن ازاحة الى اليمين كأن رأس السهم تشير الى اتجاه الازاحة .

المشروع الثانى عشر

فكرة المشروع :

مفتاحين احدهما يعمل ازاحة لليدات الى اليسار والاخر الى اليمين .

الكود:

عند الضغط على المفتاح c0 يقوم بعمل ازاحة الى اليسار وعند الضغط على المفتاح C1 يقوم بعمل ازاحة الى اليمين

```
void main() {
 trisb=0;                //portb is output

trisc=255;               //portc is input
portb=0b00000001;
while (1){
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=portb<<1;
while (portc.f0==0){};
}
}
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=portb>>1;
while (portc.f1==0){};
}
}
}
}
```

لكن يوجد مشكلة فى هذا الكود فى البداية يكون PORTB=0B00000001; واذا قمنا بالضغط على المفتاح الذى يقوم بعمل ازاحة لليمين سيصبح PORTB=0B00000000; وايضا اذا كان portb=0b00000000; وقمنا بعمل ازاحة الى اليسار سيصبح portb=0b100000000;

اذا لابد من وضع جملة تختبر هل PORTB=0B00000001; فاذا تم الضغط على المفتاح c1 فان portb=0b100000000; وايضا عندما يكون portb=0b100000000; وتم الضغط على المفتاح C0 فان PORTB=0B00000001;

سيصبح الكود هكذا

```
void main() {
 trisb=0;                    //portb is output
trisc=255;                   //portc is input
portb=0b00000001;
while (1){
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=portb<<1;
while (portc.f0==0){};
}
}
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=portb>>1;
while (portc.f1==0){};
}
}
if (portb=0b00000001){
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=0b10000000;
while (portc.f1==0){};
}
}
}
if (portb=0b10000000){
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=0b00000001;
while (portc.f0==0){};
}
}
}
}
}
```

لكن الحل غير كافى لان portb فى بداية البرنامج يساوى portb=0b00000001; وبالتالى تتحقق جملة if التى تختبر هذا الشرط وهى if (portb=0b10000000) وداخل جملة if هذة جملة if أخرى تختبر هل تم الضغط على المفتاح أم لا فاذا لم يتم الضغط على المفتاح تنتقل الى الاوامر التى تلى هذا الامر لان الميكرو ينفذ الاوامر فى غاية السرعة فبالتالى لا ينفذ الامر المرغوب لذا نضع شرط انه يفضل يلف فى لوب لحد ما اضغط ع المفتاح وهذا الامر هو while (portb.f0==1){}; ليصبح الكود بهذا الشكل :

```
void main() {
trisb=0;                    //portb is output
trisc=255;                   //portc is input
portb=0b00000001;
while (1){
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=portb<<1;
while (portc.f0==0){};
}
}
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=portb>>1;
while (portc.f1==0){};
}
}
if (portb==0b00000001){
while (portc.f1==1){};
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=0b10000000;
while (portc.f1==0){};
}
}
}
```

```
if (portb==0b10000000){
while (portc.f0==1){};
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=0b00000001;
while (portc.f0==0){};
}
}
}
}
}
```

لكن يوجد مشكلة بهذا البرنامج انه portb=0b00000001; فى بداية البرنامج لذا سيذهب لشرط if لو ضغطت على المفتاح co مش هيعمل حاجه لحد ما اضغط على c1 عندما نضغط على المفتاح C1 يصبح portb=0b10000000; لو ضغطت بعدها على c1 مش هيعمل حاجه لحد ما اضغط على c0 لو ضغطت على co هيصبح portb=0b00000001; واذا ضغطت بعدها على c1 وهكذا فهيفضل اللى ينور الليد الاولى والليد الاخيرة طب ايه الحل

الحل :

عندما يكون portb=00000001; ندخلة فى شرط اذا كان portb=00000001; افضل لف فى لوب لحد ما اى مفتاح يتم الضغط علية وذلك عن طريق الامر

while (portc.f0==1 && portc.f1==1) {}; وقولة اذا ضغط على ال c1 خلى portb=0b10000000; اذا ضغطت على c0 خلى portb=0b00000010;

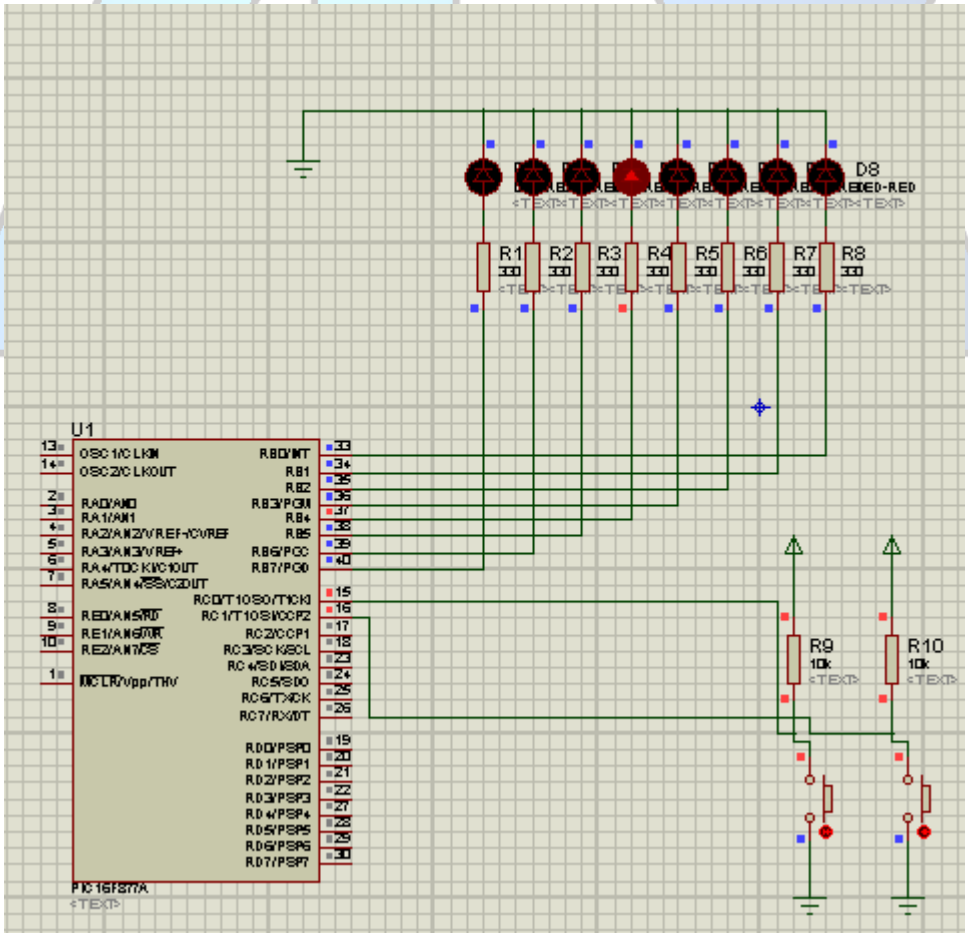
وكذلك عندما يكون portb=10000000; ندخلة فى شرط اذا كان portb=0b00000001; افضل لف فى لوب لحد ما اى مفتاح يتم الضغط عليه وقولة اذا ضغط على المفتاح c0 خلى portb=0b00000001; واذا ضغطت على c1 خلى portb=0b01000000; وبكده تكون المشكلة اتحلت .

الكود بعد التعديل

```
void main() {
trisb=0;                    //portb is output
trisc=255;                   //portc is input
portb=0b00000001;
while (1){
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=portb<<1;
while (portc.f0==0){};
}
}
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=portb>>1;
while (portc.f1==0){};
}
}
if (portb==0b00000001){
while (portc.f1==1 &&portc.f0==1){};
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=0b10000000;
while (portc.f1==0){};
}
}
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=0b00000010;
while (portc.f0==0){};
}
}
}
```

```
if (portb==0b10000000){
while (portc.f0==1 && portc.f1==1){};
if (portc.f0==0){
delay_ms(10);
if (portc.f0==0){
portb=0b00000001;
while (portc.f0==0){};
}
}
if (portc.f1==0){
delay_ms(10);
if (portc.f1==0){
portb=0b01000000;
while (portc.f1==0){};
}
}
}
}
}
```

توصيل الدائرة :



Generating sound

فى هذة المحاضرة سنتعرف على ال PIC مع بعض العناصر الكهربية الاخرى كال buzzer

عندما نريد أن نتحكم في أي جهاز أو قطعة الكترونية أو كهربية فإننا نحتاج لمعرفة نقطتين رئيسيتين :-

١- الجهد الذي يحتاجه هذا الجهاز (متردد أم مستمر) وما هو مقداره .

٢- التيار الذي يحتاجه لكي يؤدي أفضل النتائج .

فصديقنا العزيز (البك) له إمكانيات محدودة حيث أنه يستطيع إخراج جهد مستمر لا يزيد عن خمسة فولت وبشدة تيار قصوى 25 ملي أمبير ، ولكن مع ذلك نستطيع باستخدام البك أن نتحكم في أجهزة تعمل بـ ٢٢٠ فولت وتيار 5000 ملي أمبير (خمسة أمبير) بل وأكثر من ذلك ولكن بمساعدة قطع أخرى مثل الترانزستور و الريلاي Relay و

المشروع الاول

فكرة المشروع:

عمل جرس لباب عند الضغط على المفتاح نسمع صوت صفير ولما نشيل ايدنا يقف الصوت

طريقة التفكير فى المشروع :

يتم اصدار الصوت بواسطة القطعة الالكترونية والمعروفة باسم ال buzzer

في هذه التجربة المطلوب تشغيل Buzzer .

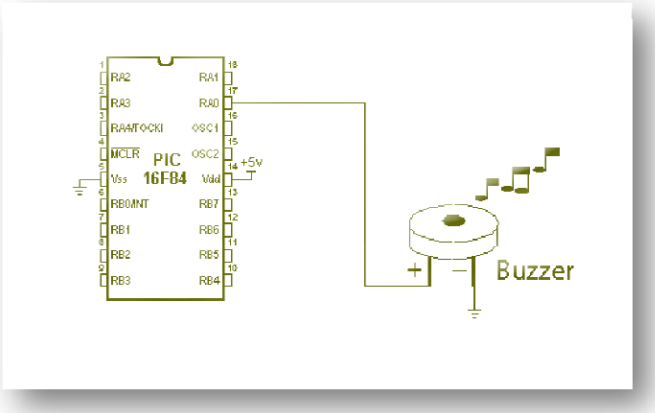
والـ Buzzer عبارة عن قطعة لها طرفان طرف موجب وطرف سالب ، هذه القطعة تصدر صوت صافرة ومن الأنواع الشهيرة ذلك النوع الذي يصدر صوت صافرة إذا طبق على طرفيه الجهد المطلوب ويسمى هذا لنوع بـ level mode Buzzer . لأنه يحتاج لمستوى معين من الجهد لكي يعمل وهذا النوع هو ما سنستخدمه في هذه التجربة . (كما يوجد قطعة أخرى تسمى Beeper تصدر صوتًا إذا طبق على طرفيها الجهد المناسب فهي تشبه الـ level Buzzer مع وجود اختلافات بسيطة) ويوجد أنواع أخرى من الـ Buzzer لا تصدر صوتاً إلا إذا تم إرسال لها نبضات (موجة مربعة) بتردد معين وهذا النوع يسمى بـ frequency mode Buzzer كما أنه بتغيير التردد يتغير الصوت الصادر من الـBuzzer

والـ Buzzer عمومًا يعمل بجهد معين وتيار معين ويمكننا معرفة الجهد والتيار من خلال الكتالوج أو datasheet أو أحيانا مما هو مكتوب على القطعة نفسها . وهذه بعض أشكاله :



إذا كان الـ Buzzer الذي سنستخدمه يحتاج إلى جهد ٥ فولت وتيار يساوي ٢٥ ملي أمبير أو أقل فإنه خاضع لقدرات البك العادية وفي هذه الحالة نقوم بتوصيله توصيل مباشر ولن نحتاج إلى قطع إلكترونية أخرى مثل الترانزستور ولهذا سنقوم بالتوصيل كما بالشكل :

انتبه لقطبية الـ Buzzer أثناء التوصيل

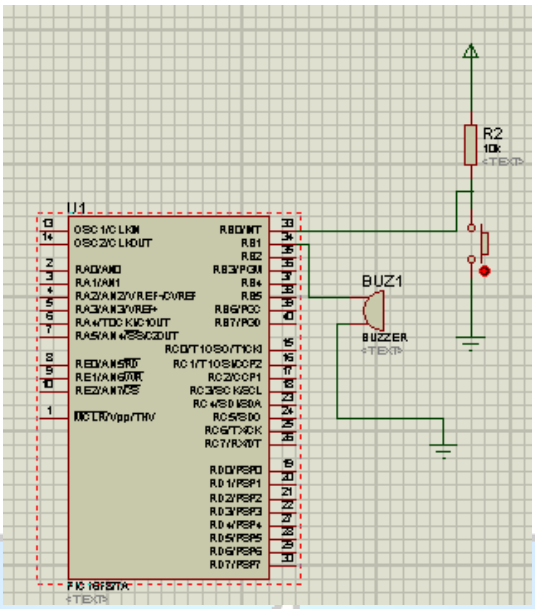


وفي هذه الحالة فإنه لكي يعمل الـ Buzzer سنقوم بتطبيق جهد موجب على الطرف الموصل به . وفي الرسم قام بتوصيل الطرف الموجب لل buzzer على رجل من رجول

الكود :

```
void main() {
trisb=0b00000001;        //rb0 is input and other pins are output
rb1_bit=0;               //initially buzzer off
for (;;){
if (rb0_bit==0){         //if button is pressed
delay_ms (10);
if (rb0_bit==0){
rb1_bit=1;             // buzzer is on
while (rb0_bit==0){};
}
}
else                   // button not pressed
  rb1_bit=0;     // buzzer is off
}
}
```

توصيل الدائرة :



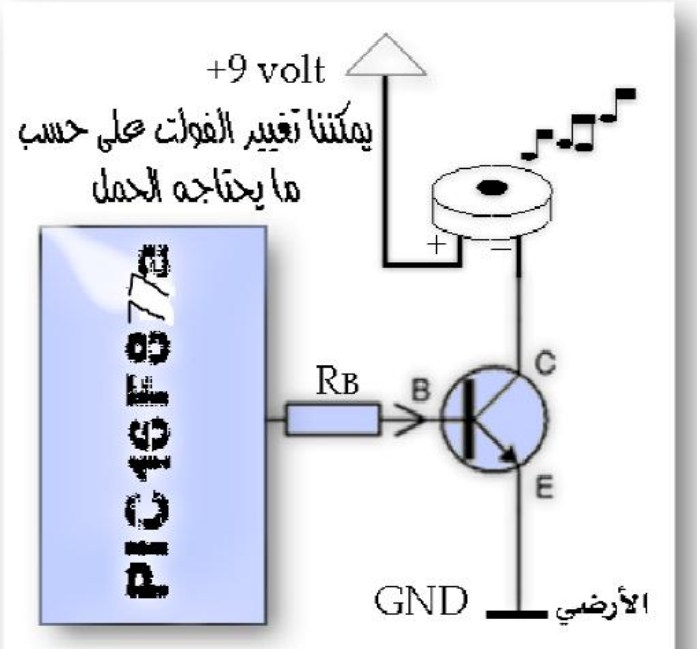
لكى يعمل ال buzzer على بروتس ويصدر صوت يتم الضغط عليه مرتين وتعديل ال operating voltage=5v

Load resistance=120

ولكن بعض أنواع ال buzzer يعمل ب 9v و 12v فما الحل ؟؟؟؟

الحل ؟؟

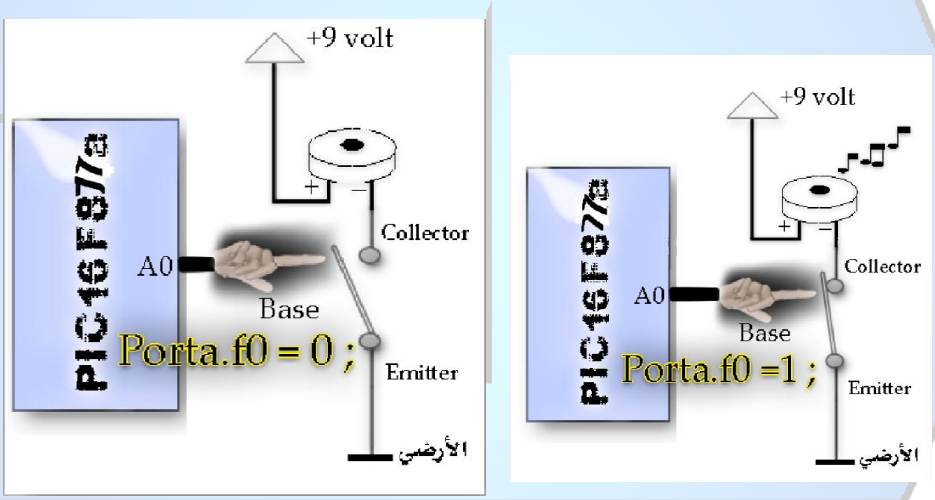
بالنسبة للكود فهو نفس الكود السابق كل ما سنغيره هو الهاردوير والحل الأسهل هنا هو استخدام الترانزستور كمفتاح . transistor as a switch كما بالشكل .



يحتاج ال buzzer الى تيار اكتر من 25 ملى امبير لكى يعمل ولكن يخرج من ال pin تيار صغير حوالى 15 ملى أمبير ولكى اشغل buzzer 9 فولت او 12 فلا يمكننى توصيل ال buzzer على رجل الميكرو مباشرة لذا اوصل على ال pin للميكرو مقاومة وترانزستور NPN ممكن bc547 حيث يدخل من ال base تيار صغير اقدراخده من الميكرو ويمر فى ال collector تيار أكبر يلائم تشغيل ال buzzer

لما تكون ال pin high يمر تيار فى الترانزستورهيصل الترانزستور الى حالة ال saturation وهيكون بين ال emitter والـ collector عبارة عن short circuit فهيوصل ارضى للدائرة فيمر تيار فى ال buzzer فيصدر صوت صفير , عندما يكون الخرج low فان الترانزستورسيصل الى مرحلة ال cutoff وهيكون بين ال emitter والـ collector عبارة عن open circuit فلا يصل ارضى للدائرة فلا يمر تيار فى ال buzzer فلا يصدر صوت .

المقاومة اللى مع الترانزستور تكون 1kΩ .



المشروع الثانى

فكرة المشروع :

نفس فكرة المشروع السابق لكن باستخدام عناصر الكترونية (speaker/buzzer/piezo) تعتمد على التردد ويتم تشغيلها بواسطة موجه مربعه .

في هذه التجربة سنتعلم كيف نصدر نغمات صوتية من السماعة speaker أو من الـ Buzzerالذي يعمل بالترددات frequency mode Buzzer أو من piezo transducer فكل هذه القطع تصدر صوتًا معينًا إذا تم إرسال إليها نبضات بتردد معين .

في مثالنا هذا سنستخدم سماعة عادية مثل سماعة الراديو التي تسمى loudspeaker أو speaker .

بعض أشكال الـ speaker



هنا لكي نصدر صوت معين سنقوم بإرسال تردد معين وليس مجرد تطبيق جهد على طرفي السماعة . ولكن كيف سنقوم بإرسال تردد معين ؟؟ ببساطة سنقوم بتطبيق جهد موجب على أحد أطراف البك ثم نجعل البك ينتظر قليلا لمدة معينة ثم نقوم بتطبيق جهد صفر على نفس الطرف ثم نجعل البك ينتظر لمدة معينة ونكرر ذلك باستمرار ، وبهذا قمنا بعمل موجة مربعة (أو أرسلنا نبضات) لها تردد معين هذا التردد يعتمد على زمن الانتظار الذي كتبناه في الكود .

فعندما نكتب الأمر portb.f0=1; ثم الأمر ; delay_ms(5) ثم نكتب الأمر portb.f0=0; ثم الأمر delay_ms(5); ونكرر ذلك باستمرار عن طريق جعل هذه الأوامر داخل جملة while(1) { } فإننا بذلك قمنا بإرسال نبضات مستمرة بتردد معين . وبتغيير الرقم خمسة الموجود في أمر الإنتظار فإننا بذلك نقوم بتغيير تردد تلك النبضات . ولكن ما هو تردد هذه النبضات في الكود السابق وكيف يتم حسابه ؟؟

الآن عندما نقوم بإرسال نبضة عن طريق الأمرين ; portb.f0=1; delay_ms(5) بهذين الأمرين قمنا بإخراج جهد موجب لمدة خمسة ملي ثانية أو بتعبير آخر فإننا قمنا بإرسال نبضة Hi أو نبضة واحد مقدارها خمسة ملي ثانية . جميل وبنفس الفكرة فإننا عندما نكتب الأمرين portb.f0=0; delay_ms(5); فإننا قمنا بتطبيق جهد صفري لمدة خمسة ملي ثانية أو بتعبير آخر فإننا قمنا بإرسال نبضة low أو نبضة صفر مقدارها خمسة ملي ثانية . إذن العملية السابقة يمكننا أن نقول عنها أننا نرسل واحد وأصفار .

أرسلنا واحد لمدة ٥ ملي ثانية ثم أرسلنا صفر لمدة ٥ ملي ثانية إذن المجموع هو ١٠ ملي ثانية ، أي أن الزمن الذي استغرقته الدورة الكاملة يساوي ١٠ ملي ثانية ومعنى زمن الدورة الكاملة أي زمن نبضة الواحد + زمن نبضة الصفر اللذان يتكرران باستمرار ، و هذا الزمن هو زمن الدورة الكاملة أو ما يسمى بالزمن الدوري .

جميل بهذا قد انتهينا من الشرح ...!!

لا لا انتظر لم ننتهي بعد ، فهدفنا هو حساب التردد وليس الزمن الدوري .!!

إذا كان لديك الزمن الدوري فبكل بساطة ستحصل على التردد يا صديقي فقط قم بقسمة الرقم واحد على الزمن الدوري الذي حسبته .

إذن التردد في هذا المثال يساوي : $\frac{1}{10 \text{ملي ثانية}}$ وطبعاً كلمة ملي تعني أننا سنضرب القيمة في 10^{-3}

إذن التردد يساوي $\frac{1}{10 \times 10^{-3}}$ = ١٠٠ هيرتز .



معلومات مهمة لابد منها : لعلك لاحظت في الأمثلة السابقة أن التردد الناتج هو تردد صغير نسبيًا فكيف نحصل على ترددات كبيرة مثل ١٠ آلاف هيرتز أو ٥٠ ألف هيرتز ونحو ذلك من الترددات الكبيرة . إذا فكرت قليلاً فستجد أنه لا يمكننا فعل ذلك باستخدام الأمر delay_ms لأنه في هذه الحالة نحن نحتاج لوحدات أقل من الملي ثانية مثل الميكرو ثانية . وللتذكير فإن الملي ثانية هو جزء من ألف جزء من الثانية . أما الميكروثانية فهو جزء من مليون جزء من الثانية ، أي أن الميكرو ثانية = $\frac{1}{1000000}$ = ١ x ١٠ $^{-6}$ ثانية

والسؤال هنا كيف أجعل البك ينتظر واحد ميكرو ثاينة ؟؟ ببساطة قم بتحويل حرف m إلى u

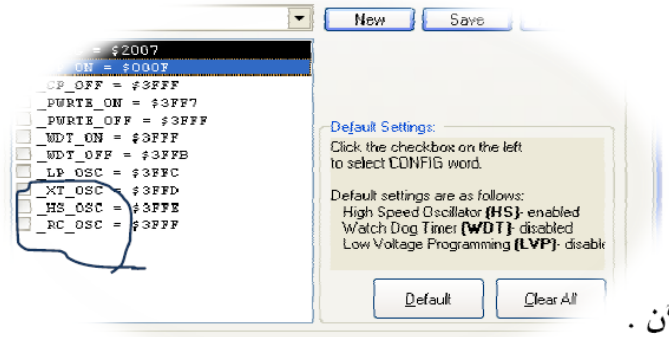
```
delay_us(1);
```

يجب أن يكون المذبذب الذي تستخدمه تردده عالي وكلما تعاملت مع زمن أصغر مثل الميكروثانية delay_us(1) عليك أن تختار مذبذب تردده كبير مثل ١٥ ميجا أو ٢٠ ميجاهيرتز مثلاً ولكن السؤال هو لماذا ؟ والسؤال الأهم أولاً هو ما هي المذبذبات ؟

ما هي المذبذبات Oscillators ؟ المذبذبات هي دائرة تنتج موجة من تيار مستمر . هذا هو أحد التعريفات العلمية للمذبذبات . إذن الوظيفة هي إنتاج موجة أو إنتاج نبضات ، وفي تطبيقاتنا هنا يتم توصيل المذبذب أياً كان نوعه بالمايكروكنترولر فيقوم المذبذب بإرسال هذه النبضات للمايكروكنترولر وكل نبضة أو مجموعة نبضات (أو بالأصح كل دورة أو مجموعة دورات cycles) تأتي للمايكروكنترولر يقوم بتنفيذ أمر معين في الكود . لهذا المذبذب ضروري جداً للبك وبدونه لا يعمل . و إذا زادت سرعة هذه النبضات (أي زاد تردد المذبذب) زادت سرعة المايكروكنترولر في تنفيذ الأوامر .

والسؤال الأهم الآن هو : ما هي الاعدادت التي ستتغير إذا استخدمنا كريستال ٢٠ ميجاهيرتز مثلا بدلاً من استخدام RC oscillator ؟

في لغة مايكروسي وعند إنشاء مشروع جديد نكتب التردد الخاص بالمذبذب الذي نستخدمه ثم سنختار واحد من الاختيارات التالية :



١- _RC_OSC
٢- _XT_OSC
٣- _HS_OSC

بالطبع هناك اختيارات أخرى لكن لن نحتاجها الآن .

إننا نختار _RC_OSC عندما نستخدم مذبذب من النوع RC oscillator

نختار _XT_OSC عندما نستخدم كريستال تردده له إحدى القيم المحصورة بين 200 كيلوهيرتز إلى 4 ميجا هيرتز .

نختار _HS_OSC عندما نستخدما كريستال تردده من 4 ميجا هيرتز إلى 25 ميجا هيرتز .

أولا : كيف نقوم بتوصيل السماعة للمايكروكنترولر ؟

لابد أولا من معرفة النقطتين الرئيسيتين : ١- الجهد التي تحتاجه السماعة ٢- التيار .

إذا كانت السماعة تعمل بجهد يساوي ٥ فولت فإننا نقوم بتوصيلها توصيل مباشر و إذا كانت السماعة تعمل بجهد أقل من ٥ فولت فإننا نضع مقاومة قبل السماعة كما كنا نفعل في توصيل الليد ، وإذا كانت تعمل بجهد أكبر من خمسة فولت أو أن التيار الذي تحتاجه أكبر من ٢٥ ملي أمبير فإننا سنحتاج إلى ترانزستور نستخدمه كمفتاح كما فعلنا مع الـ Buzzer.

ملحوظة : بعض أنواع السماعات تحتاج إلى أن نضع قبلها مكثف coupling capacitor مثل سماعات الأذن

ويستخدم المكثف لعمل coupling لل DC لكى تدخل على السماعة موجة مربعة نقية .

لاحظ أنه مع تغير التردد فان الصوت الصادر سيتغير فعند زيادة التردد تزداد حدة الصوت .

الكود :

```
 void main() {
trisb=0b00000001;       //rb0 is input and other pins are output
rb1_bit=0;              //initially buzzer off
for (;;){
if (rb0_bit==0){        //if button is pressed
delay_ms (10);
while (rb0_bit==0){ // while button is pressed do rhat
rb1_bit=1;            // buzzer is on , high pulse
delay_ms (5);        // wait with high pulse
rb1_bit=0;          // buzzer is off , low pulse
  delay_ms (5);     // wait with low pulse
   }
while (rb0_bit==0){};
}

  rb1_bit=0;    // buzzer is off   when switch isnot pressed
}
}
```

Tones

والآن هيا بنا إلى رحلة ممتعة وهي رحلة إنتاج النغمات ، الآن عزيزي القارئ أنت مهيأ تماماً لصنع نغمات صوتية أو موسيقية مختلفة لا ينقصك إلا معلومة واحدة بسيطة جداً وهي :

لإنتاج نغمة صوتية فإننا نقوم بإصدار تردد معين لمدة بسيطة ثم نقوم بإصدار تردد أخر لمدة مثل السابقة أو مختلفة عنها ... وهكذا .

والنغمة الصوتية الناتجة تعتمد على شيئين :

١- الترددات المستخدمة .

٢- مدة تكرار الموجة (يعتمد على الرقم الذي نضعه في جملة for)

سأوضح لك الأمر نحن الآن نقوم بجعل البك يقوم بإنتاج تردد معين لمدة معينة ، وهذا التردد لكي نغيره فإننا نقوم بتغيير الرقم داخل أمر الإنتظار ... جميل ،، ما رأيك أن نقوم باستبدال هذا الرقم ونضع بدلاً منه متغير هذا المتغير تتغير قيمته باستمرار بأن تتزايد مثلا أو تتناقص وبالتالي سيتغير الصوت لتغير التردد .

والسؤال هنا هل يمكننا أن نكتب الأمر delay_ms(x); ؟ حيث أن x متغير تم تعريفه من قبل ؟؟؟ هل يمكننا فعل ذلك وجعل ذلك داخل جملة while(1) لكي يتكرر هذا الأمر باستمرار ونضع أيضاً الأمر

```
If (x=0; x<100; x++) {delay _ ms(x) ;}
```

أليست فكرة جميلة ؟؟ في كل مرة ستتغير قيمة المتغير x فمرة تكون بواحد وكأننا كتبنا الأمر delay_us(1) ومرة تكون باثنين وكأننا كتبنا delay_us(2) وهكذا كل مرة تزداد مدة الإنتظار . وبالتالي سيتغير الصوت الناتج .. حقاً إنها فكرة جميلة ، لكنها طريقة فاشلة للأسف ..!!

لماذا ؟ لأنه إذا كتبنا الكود السابق فلن تستطيع لغة مايكروسي ترجمته وتظهر رسالة خطأ بالأسفل . . ولكن لماذا ؟ لأن أمر الانتظار سواءاً كان بالميكروثانية أو بالملي ثانية لا يمكن أن نضع داخله متغير . هكذا صُمّمت لغة البر مجة مايكروسي ، فيمكننا أن نضع بين القوسين ثابت رقمي فقط كما تعودنا ولا يمكنا وضع متغير داخل القوسين.

ببساطة عندما تريد تطبيق هذه الفكرة في أي مشروع من مشاريعك قم بكتابة الأمر ;delay_us(1) وقم بتكرار هذا الأمر عدة مرات على حسب قيمة x . بمعنى أننا سنضع هذا الأمر داخل جملة for بشكل معين كما يلي :

```
for(y=0;y<x;y++) delay_us(1);
```

وعلى حسب قيمة x سيكون زمن الانتظار .

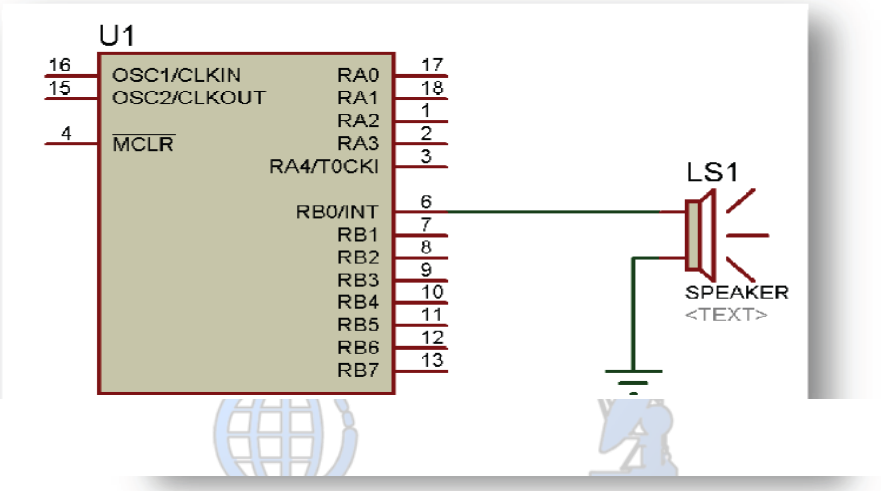
لكن هذة الطريقة تصدر نغمات عشوائية لذا اليك بعد الاكواد المجربة لبعض النغمات .

المشروع الثالث

فكرة المشروع :

أولاً : إليك بعض النغمات التي يمكنك استخدامها في تطبيقات أجهزة الإنذار ، فهذا الصوت الذي سنولده يشبه صوت سيارات الشرطة أو الاسعاف .

سنقوم بتوصيل السماعة (في الحقيقة أو في برنامج المحاكاة)



ثم نقوم بإنشاء مشروع جديد نستخدم فيه مذبذب من النوع crystal (كريستال) ذو تردد 4MHz ونختار _XT_OSC أو _HS_OSC . ثم نقوم بكتابة الكود التالي :

```
void main(){ int x; trisb=0; portb=0;
while(1){
    for(x=0;x<400;x++){ portb.f0=1;  delay_us(750);
                        portb.f0=0;  delay_us(800);}
    for(x=0;x<450;x++){ portb.f0=1;  delay_us(500);
                         portb.f0=0;  delay_us(750);}
        }   }
```

المشروع الرابع

هل تتذكر ذلك الصوت الذي نسمعه من بعض مسدسات الأطفال ؟ إليك هذه النغمة الشبيهة بما أقصد:

```
void main()
{ int x,y;  trisb=0;   portb=0;
while(1){
        for(x=0;x<50;x++){
            portb.f0=1;  for(y=0;y<x;y++)delay_us(10);
            portb.f0=0; for(y=0;y<x;y++)delay_us(10);
                              }
        }
}
```

وهنا يجدر الإشارة أنه فيما يخص أمر الانتظار بالملي ثانية delay_ms ذكرنا عنه فيما سبق هو والأمر delay_us أنه لايمكننا أن نضع بين القوسين متغير ويجب وضع رقم . ولكن المعلومة الجديدة أنه يوجد أمر في لغة مايكروسي يمكننا من عمل انتظار بالملي ثانية على حسب قيمة متغير هذا الأمر اسمه ()vdelay_ms

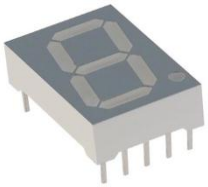
```
vdelay_ms(yy);
```

Chapter (5)

7-Segment



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

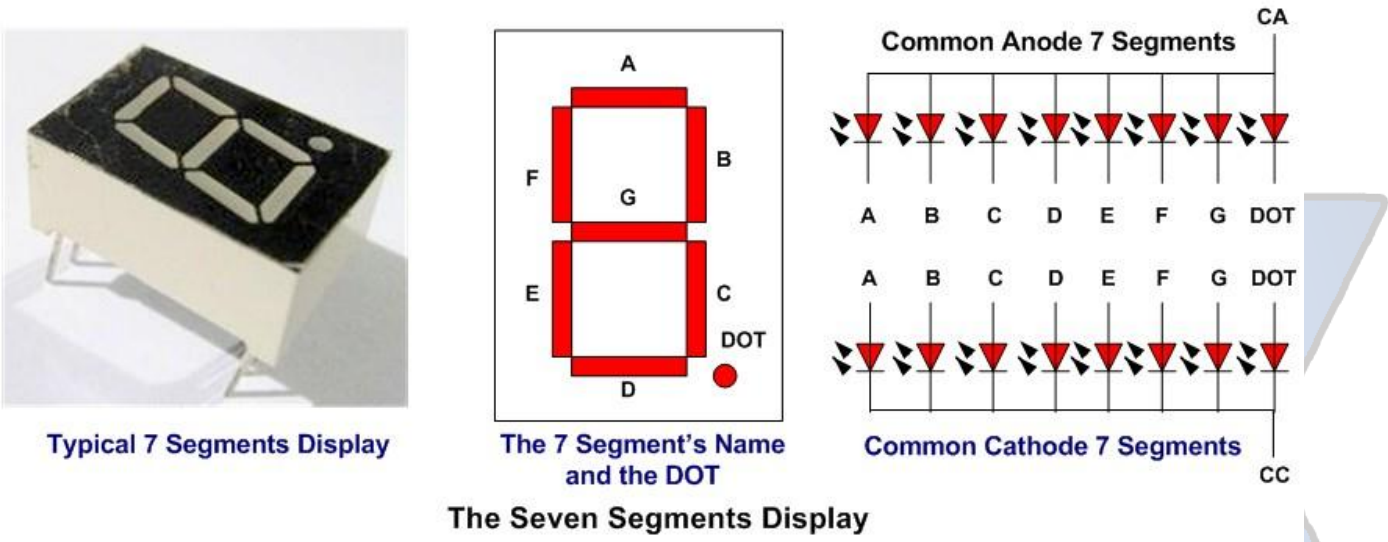


Seven segment

السفن سيجمنت ما هي إلى سبعة ليدات تم توصيلهم بطريقة معينة بحيث يمكن من خلالها تمثيل الأرقام من صفر إلى تسعة بكل سهولة عن طريق إضاءة بعض الليدات وإطفاء البعض .

والشكل التالي يوضح الحرف أو الرقم الدال على كل ليد في السفن سيجمنت ويوضح أنه عبارة عن ليد .

http:/www.ermicro.com/blog



The Seven Segments Display

فإذا أردنا أن نظهر الرقم واحد فعلينا أن نضيء الليد b و c ونطفئ الباقي . وإذا أردنا أن نظهر الرقم سبعة فسنضيء الليدات a و b و c ونطفئ الباقي .. وهكذا الحال في باقي الأرقام . ويجدر الاشارة إلى أنه في بعض أنواع السفن سيجمنت يوجد ليد آخر اسمه d.p وهو اختصار لكلمة decimal point أي العلامة العشرية فإذا أردنا مثلا تمثيل الرقم ٢٣,٥ فإننا سنحتاج لثلاثة سفن سيجمنت الأولى نعرض عليها الرقم اثنين والثانية الرقم ثلاثة مع إظهار العلامة العشرية والثالثة نظهر عليها الرقم خمسة .

من الرسم السابق يتضح أن الليدات تم توصيلها بطريقة معينة بحيث أن المصعد (الأنود anode) لكل ليد موصل بالآخر ويسمى هذا النوع من السفن سيجمنت بالنوع common anode أي مصعد مشترك .

ويوجد أنواع آخرى من السفن سيجمنت يسمى common cathode أي مهبط مشترك وفي هذه الحالة يكون كل مهبط (كاثود cathode) يتم توصيله بالمهبط الآخر ويسمى هذا النوع من السفن سيجمنت بالنوع common cathode أي مهبط مشترك .

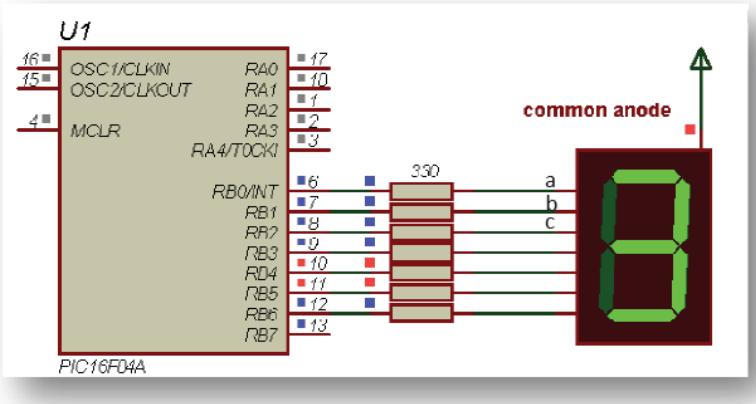
مما سبق يتضح أن السفن سيجمنت لها سبعة أطراف (طرف لكل ليد) بالإضافة لطرف آخر للعلامة العشرية بالإضافة للطرف المشترك common . (بعض السفن سيجمنت يكون بها طرفان common يمكنك استخدام أي منهما ولا فرق بينهما) .

السؤال المهم الآن : ما هو الفرق بالنسبة لنا عندما نستخدم سفن سيجمنت من النوع common anode أو common cathode ؟؟

ببساطة عند استخدام سفن سيجمنت من النوع common cathode فإن الطرف الـ common (المشترك) يتم توصيله بالسالب ولإضاءة أي ليد وليكن f مثلا فإننا نطبق على هذا الطرف جهد موجب أي واحد منطقي .

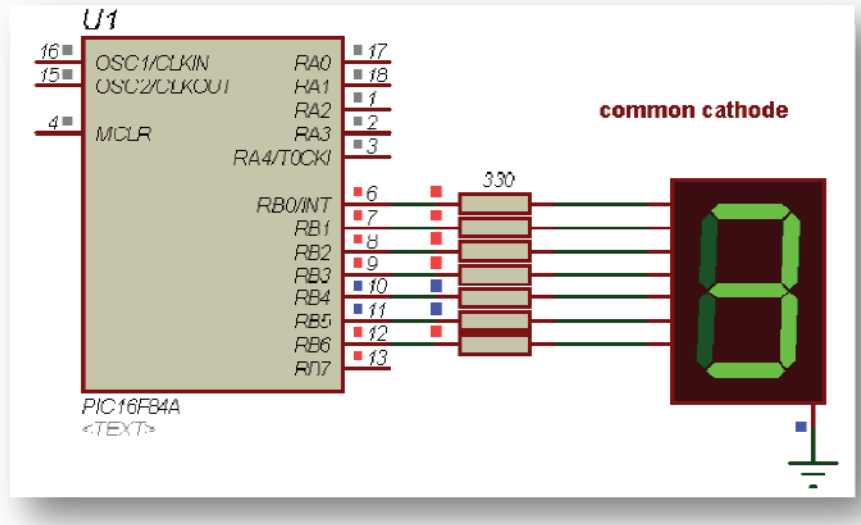
أما عند استخدام سفن سيجمنت من النوع common anode فإن الطرف الـ common (المشترك) يتم توصيله بالموجب ولإضاءة أي ليد وليكن f مثلا فإننا نطبق على هذا الطرف جهد سالب أي صفر منطقي .

وإليك الآن مثال لكيفية إظهار الرقم ثلاثة مرة على السفن سيجمنت من النوع common anode ومرة على النوع common cathode .




```
void main()
{
trisb=0;
portb=0b00110000;
}
```

نلاحظ أن الليد المنطفئ أشرنا إليه في الكود بواحد ونلاحظ أيضا أن الطرف B7 غير مستخدم فلن تؤثر قيمته سواءاً كانت بصفر أو واحد .



Common cathode:

هنا وصلنا الطرف المشترك

بالسالب والكود هو نفس السابق

مع جعل الصفر واحد

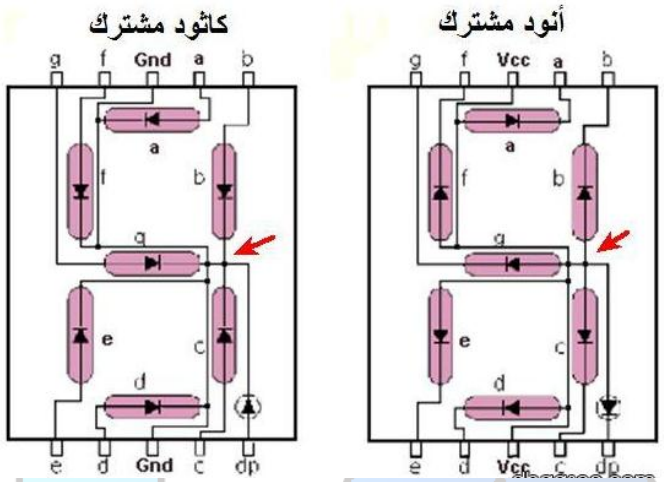
وجعل الواحد صفر . ويمكن

عمل ذلك بطريقتين :-

```
void main()
{
trisb=0;
portb=~0b00110000;
}
```

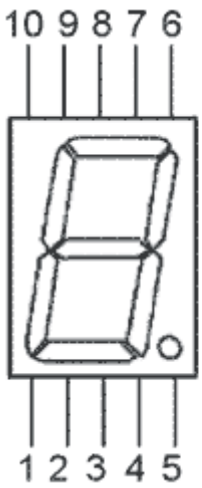
أو

```
void main()
{
trisb=0;
portb=0b11001111;
}
```



كيف أعرف أن الطرف كذا من السفن سيجمنت هو الطرف رقم a أو b أو ... وذلك عمليا وفي برنامج بروتس ؟

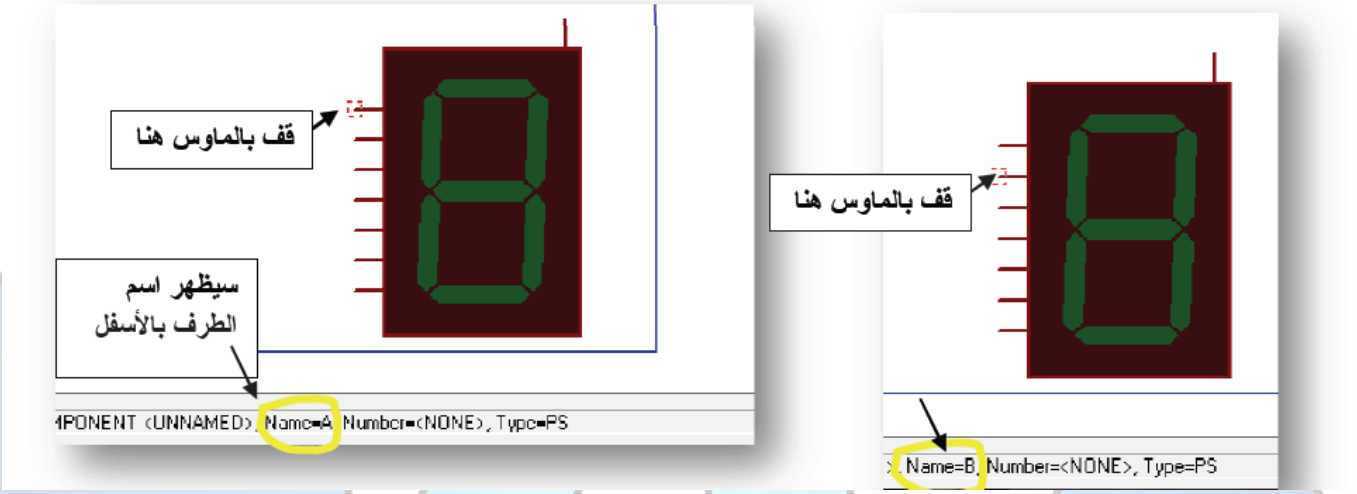
عمليا عن طريق الافو فى ال common cathode نضع الطرف الاسود من الافو على الرجل التى فى منتصف ارجل ال 7 segment فهى دائما تكون الطرف ال common ونضع الطرف الاحمر من الافو على أى رجل من رجول ال 7 segment مع ملامسة أى رجل من رجول ال segment بالطرف الاحمرتضئ ليد معينة فتكون هذة الرجل تشير الى الطرف رقم a , b,.... والذى تعبر عنه الليد والمرقمة بالصورة التى بأعلى ليصبح الاطراف بعد الاختبار كما فى الصورة :



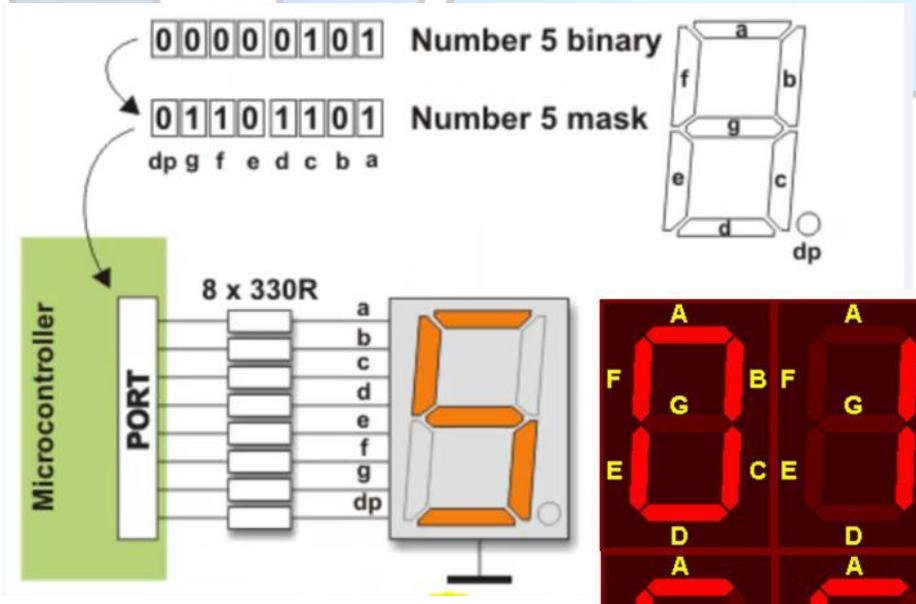
PIN NO.	E MAN6960
1	Cathode E
2	Cathode D
3	Com. Anode
4	Cathode C
5	Cathode D.P.
6	Cathode B
7	Cathode A
8	Com. Anode
9	Cathode F
10	Cathode G

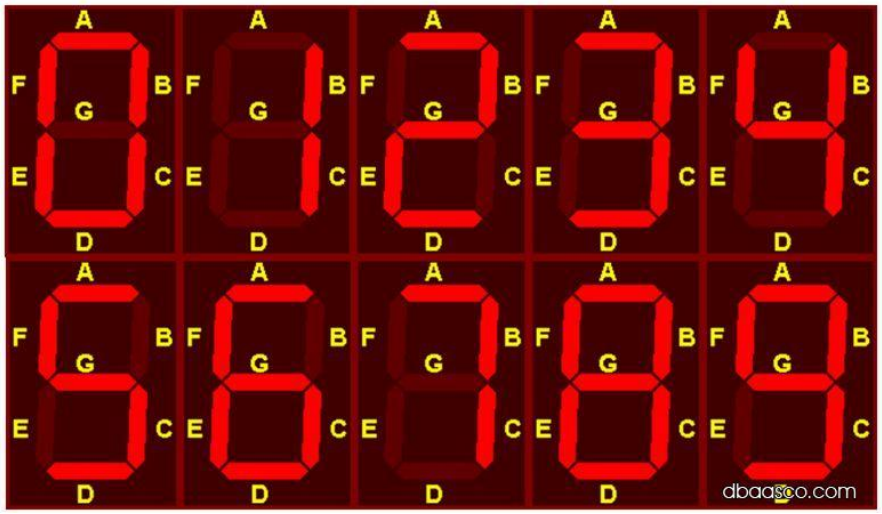
برنامج المحاكاة isis فستقف بالماوس (الفأرة) على الطرف الخاص بالسفن سيجمنت المراد معرفة رقمه . فسيظهر لك في الشريط الأسفل من البرنامج (شريط الحالة status bar) رقم العنصر كما هو واضح بالصور التالية :

وعليك أن تلاحظ عند التجربة في برنامج المحاكاة أنه قد يسمى الليدات بأسماء أخرى غير التسمية الأساسية ، فربما يجعل الليد صاحب الرقم أو الحرف f هو الليد g .



أو الاطراف كما فى الصورة :





المشروع الاول

البرنامج :

عبارة عن عداد تصاعدى up counter

ممكن استخدم الطريقة دى اكتب كل رقم واعمل delay بعدة واعرض اللى بعدية

```
void main() {

trisb=0;

for(;;){

        portb=0b00111111;  delay_ms(1000);

        portb=0b00000110;  delay_ms(1000);

        portb=0b01011011;  delay_ms(1000);

        portb=0b01001111 ;  delay_ms(1000);

        portb=0b01100110 ;  delay_ms(1000);

        portb=0b01101101 ;  delay_ms(1000);

        portb=0b01111101;  delay_ms(1000);

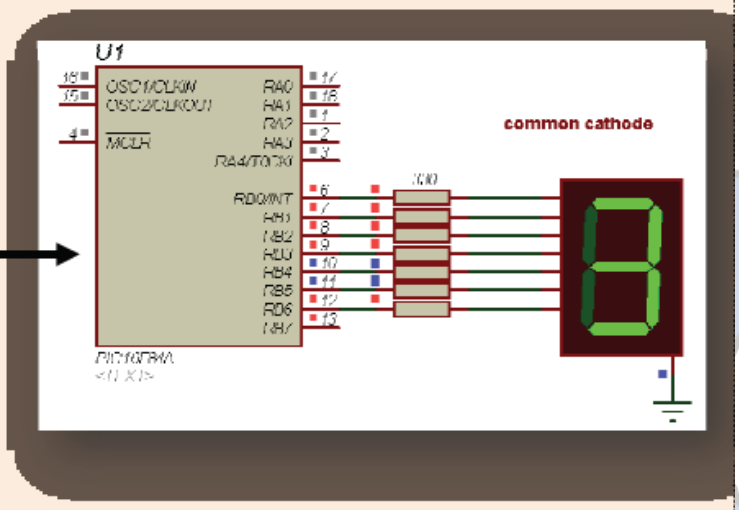
        portb=0b00000111;  delay_ms(1000);

        portb=0b01111111 ;  delay_ms(1000);

        portb=0b01101111 ;  delay_ms(1000);

    }

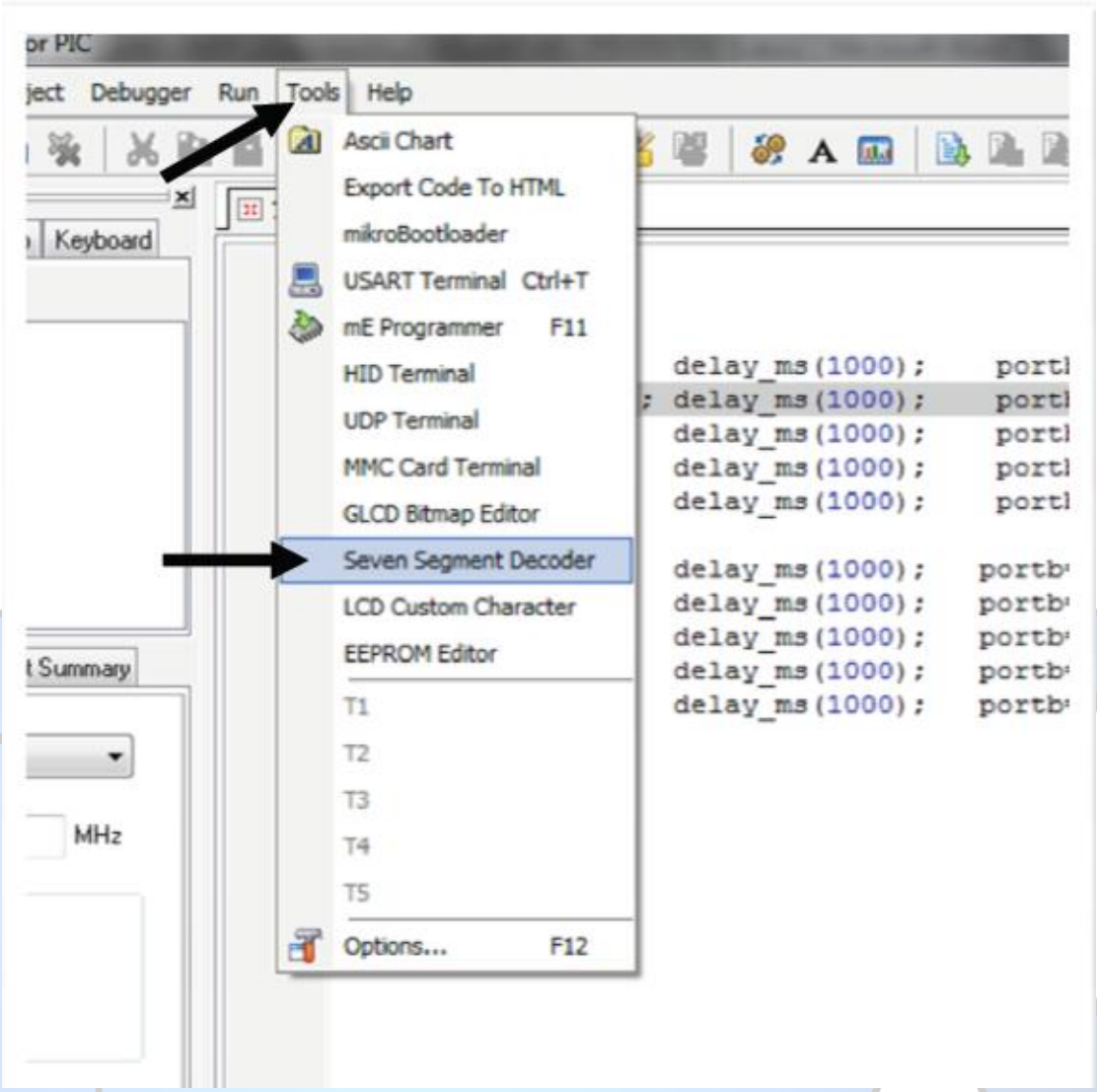
}
```



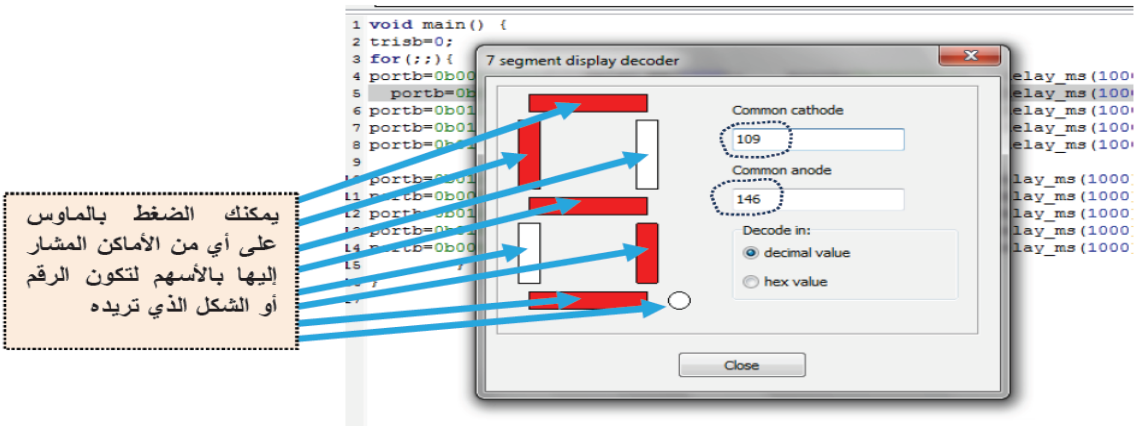
نشير إلى معلومة مفيدة وهي إمكانية موجودة في لغة مايكروسي .

هذه الإمكانية تسهل علينا كتابة الكود الذي يظهر الرقم الذي نريده على السفن سيجمنت

اختر قائمة tools ثم اختر seven segment decoder كما بالصورة التالية :



ستظهر لك الشاشة التالية :



ارسم الشكل الذي تريده وسيظهر لك رقمين رقم خاص بالسفن سيجمنت من النوع common cathode

ورقم خاص بالنوع common anode فمثلا لقد تم رسم الرقم خمسة ولنعرض الرقم خمسة سنكتب الكود التالي portb=109; هذا إذا كنا نستخدم سفن سيجمنت common cathode أما إذا كنا نستخدم النوع الآخر فسنكتب portb=146; . وهذه الأداة بالطبع توفر الكثير من الوقت .

من الممكن إنشاء متغير x قيمته تزداد ومع كل قيمة ينفذ أمر معين مع ملاحظة أننا نعرف x فى البداية على أنها تساوى صفر :

وبكده وفرنا كتابة الامر delay 8 مرات

```
void main() { char x;

trisb=0;

for(;;){

        if(x==0) portb=0b00111111;

        if(x==1) portb=0b00000110;

        if(x==2) portb=0b01011011;

        if(x==3) portb=0b01001111;

        if(x==4) portb=0b01100110;

        if(x==5) portb=0b01101101;

        if(x==6) portb=0b01111101;

        if(x==7) portb=0b00000111;

        if(x==8) portb=0b01111111;

        if(x==9) portb=0b01101111;

    delay_ms(1000);

        X++;

       If(x==10)x=0;

    }
```

لكن هذة الطريقة غير عملية وخصوصا لو بعمل مشروع ساعة أو counter

الطريقة الاحسن باستخدام الامر (case) switch

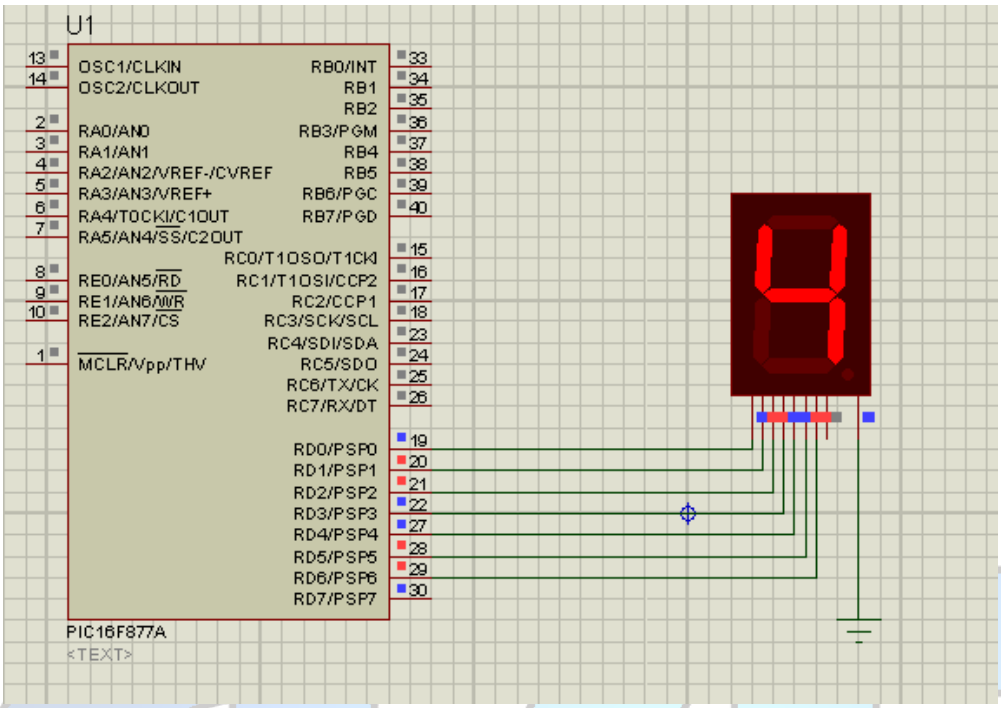
هنعمل function ننادى عليها نديها input وترجعلنا ب output هنسميها count وهنعرف متغير اسمه up له قيمة ابتدائية ب zero وكل ثانية يزيد بمقدار 1 مع كل عدة يروح لل function المسماة ب count اخد الرقم ال binary اللى بينورلى الليدات الموجودة فى السجمنت والتى تكافئ هذا الرقم مثلا عند up=0 هيروح للدالة count ويرجع ب b00111111 0واللى بخرجها على portb فبتخلى السجمنت تعرض رقم zero وهكذا

والامر case..... switch هيختبر لو عندى المتغير up=2 يقوم بتنفيذ أمر معين لو يساوى 4 هينفذ أمر تانى وهكذا .

```
unsigned short count (unsigned short num); // define function at the begining of the program
  unsigned char up=0;
void main() {
trisd=0;      //portd is output
portd=0;      // put zero at portd as initial value
for (;;){     //infinite loop
portd=count (up);        //call function count
up++;
delay_ms(1000);
if (up==10) up=0;
}
}
  unsigned short count (unsigned short num){
  switch (num){
  case 0 :return 0b00111111;
    case 1 :return 0b00000110;
      case 2 :return 0b01011011;
      case 3 :return 0b01001111;
         case 4 :return 0b01100110;
             case 5 :return 0b01101101;
               case 6:return 0b01111101;
          case 7 :return 0b00000111;
        case 8 :return 0b01111111;
         case 9 :return 0b01101111;
         }
         }
```

فى الاول up=0 هينادى على ال function count هيروح ل count وهيخلى قيمة up=0 =num هيروح ل switch (0) فهيختار case 0 ويرجع بالقيمة المخزنة وهى ob00111111; ويخرجها على portb فيظهر على السجمنت رقم zero وبعدين تزيد up بمقدار 1 فتصبح up=1 ثم ينتظر لمدة تانية ويشوف هل up==10 لا فيروح ينادى على count ويرجع ب case1 فيظهر رقم 1 على السجمنت وهكذا ال ان تساوى up=10 فى الوقت دا up=0 ويبدأ يعد من جديد وهكذا .

توصيل الدائرة :



المشروع الثانى

البرنامج :

عداد تنازلى down counter .

الكود :

لانه عداد تنزلى بدأنا up بقيمة ابتدائية 10 وتقل بمقار 1 كل ثانية وعندما تصل up الى قيمة 0 فاننا نجعلها تساوى 10 ليبدأ العد من جديد لاحظ انه تم تقديم خطوة انقاص up بمقدار واحد عن خطوة ال function call على عكس البرنامج السابق وذلك لانه اذا لم يتم عمل ذلك فلن يعرض رقم zero على ال segment وذلك لانه عند الرقم 1 سيظهر على السجمنت هيقل up بمقار واحد فتصبح زيرو يتحقق الشرط وتصبح up بمقدار 9 دون ظهور zero على السجمنت لذا تم تقديم هذة الخطوة كما هو مبين فى الكود .

```
  unsigned short mask (unsigned short num);
  unsigned char up=10;
void main() {
trisd=0;      //portd is output
portd=0;      // put zero at portd as initial value
for (;;){
  up--;
portd=mask (up);
delay_ms(1000);
if (up==0) up=10;
}
}
  unsigned short mask (unsigned short num){
  switch (num){
  case 0 :return 0b00111111;
    case 1 :return 0b00000110;
      case 2 :return 0b01011011;
      case 3 :return 0b01001111;
         case 4 :return 0b01100110;
             case 5 :return 0b01101101;
               case 6:return 0b01111101;
          case 7 :return 0b00000111;
        case 8 :return 0b01111111;
         case 9 :return 0b01101111;
         }
         }
```

نفس توصيل الدائرة السابقة .

المشروع الثالث

البرنامج :

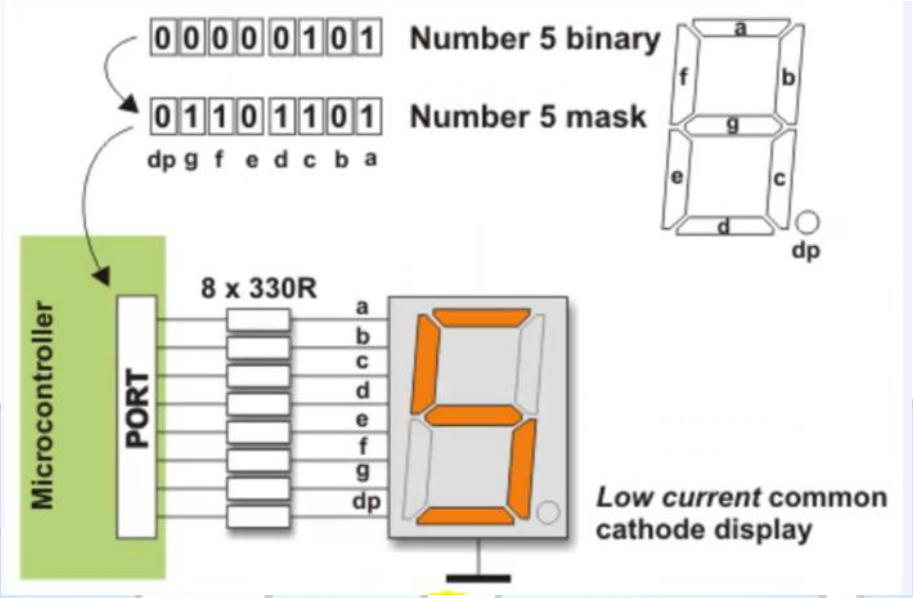
هو عبارة عن عداد زوجى even counter

الكود:

```
unsigned short count (unsigned short num); // define function at the begining of the program
  unsigned char up=0;
void main() {
trisd=0;      //portd is output
portd=0;      // put zero at portd as initial value
for (;;){     //infinite loop
portd=count (up);        //call function count
up=up+2;                  //increment by 2
delay_ms(1000);         // delay 1 sec
if (up==10) up=0;            //when up=10 count again
}
}
  unsigned short count (unsigned short num){
  switch (num){
  case 0 :return 0b00111111;
    case 1 :return 0b00000110;
      case 2 :return 0b01011011;
      case 3 :return 0b01001111;
         case 4 :return 0b01100110;
             case 5 :return 0b01101101;
               case 6:return 0b01111101;
          case 7 :return 0b00000111;
        case 8 :return 0b01111111;
         case 9 :return 0b01101111;
         }
         }
```

الكود مش هيختلف عن ال up counter الا ان العد هيزيد بمقار 2 فهستبدل الامر ++up ب up=up+2; فقط

نلا حظ فى هذة الرسمة وجود مقاومات قبل ال seven segment



ستلاحظ إضافة مقاومات قبل السفن سيجمنت . فمتى نضيف هذه المقاومات ومتى لا نضيفها ؟؟ عند تجربة الدائرة في برنامج المحاكاه بروتس فإنه لن يفرق ذلك سواءاً وضعت المقاومات أم لا لأن العناصر في هذا البرنامج غير قابلة للتلف. ولكن عند تطبيق البرنامج عمليا وفي الواقع فيجب أن تعلم نوع السفن سيجمنت المستخدمة فهناك أنواع تعمل بـ خمسة فولت (والبك يخرج 5 فولت) لذلك يمكن توصيلها مباشرة دون مقاومات . وهناك أنواع أخرى من السفن سيجمنت تعمل بجهد أقل من 5 فولت لذلك يجب أن نضع قبلها مقاومات لتقليل الجهد ولكي لا تتلف السفن سيجمنت .

المشروع الرابع

فكرة البرنامج :

عمل برنامج مكون من مفتاحين عند الضغط على المفتاح الاول يقوم بالعد التصاعدى عند الضغط على المفتاح الثانى يقوم بالعد التنازلى .

الكود :

```
unsigned short count (unsigned short num); // define function at the begining of the program
signed char up=0;
void main() {
trisd=0; //portd is output
trisc=255; //portc is input
portd=0; // put zero at portd as initial value
for (;;){ //infinite loop
if (rc0_bit==0){

delay_ms (10);
if (rc0_bit==0){

up++; //increment by one
if(up>=10) up=0 ;
portd=count (up); //call function count

while (rc0_bit==0){}
delay_ms (10); // delay 1 sec

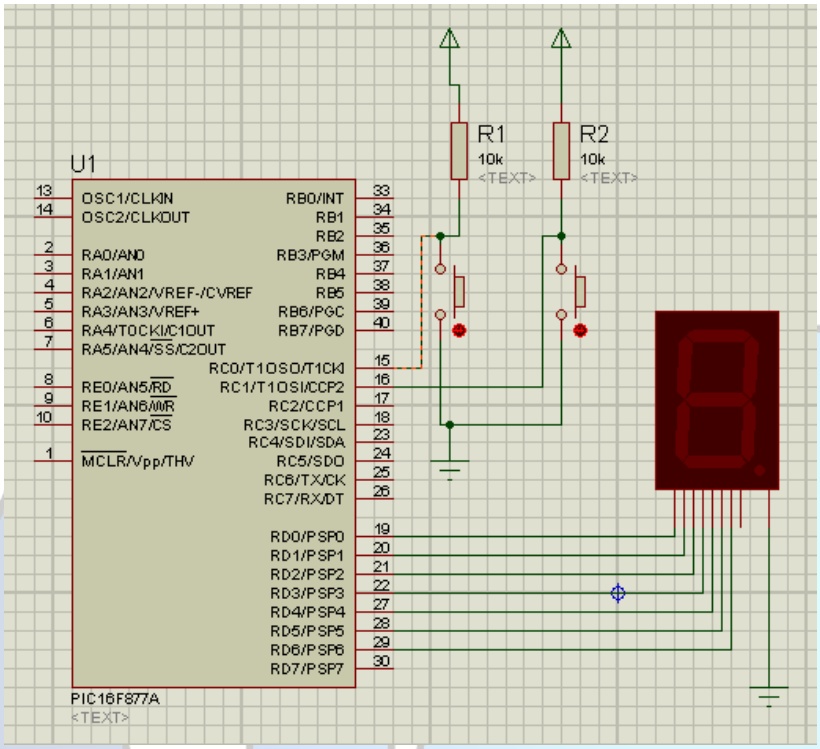
}
}
if (rc1_bit==0){
delay_ms (10);
if (rc1_bit==0){
up--;
if(up<0){ up=9; }
portd=count (up);
while (rc1_bit==0){}
delay_ms (10); // delay 1 sec
}
}
}
}
```

```
unsigned short count (unsigned short num){
switch (num){
case 0 :return 0b00111111;
case 1 :return 0b00000110;
case 2 :return 0b01011011;
case 3 :return 0b01001111;
case 4 :return 0b01100110;
case 5 :return 0b01101101;
case 6:return 0b01111101;
case 7 :return 0b00000111;
case 8 :return 0b01111111;
case 9 :return 0b01101111;
}
}
```

فنلاحظ أنه تم تعريف ال call function فى بداية البرنامج قبل ال main وتم تعريف المتغير up على أنه من النوع signed وذلك حتى يشمل الاعداد الموجبة والسالبة للسبب الذى سنذكره بعد قليل فالمتغير up له قيمة ابتدائية بصفر وعند الضغط على المفتاح الاول تزداد قيمة up لتصبح واحد ثم يختبر up هل هى أكبر من 10 أم لا فان كانت أقل من 10 يذهب لل call function وياخذ منها القيمة اللى هيخرجها على رجول الميكرو والتى تعرض الرقم الموجود فى up وهكذا حتى تصير up=10 يختبر الشرط يلقيه اتحقق يجعل قيمة up=0 ويذهب لل call function ويظهر رقم صفر على السجمنت فاذا قمنا بالضغط بعدها على المفتاح الثانى يقلل قيمة up بمقدار واحد فتصبح ب 1- فيجد أن الشرط لو up<0 تحقق فيجعل قيمة up=9

ثم يذهب لل call function ويعرض القيمة 9 على السجمنت ومن هنا عرفنا لماذا تم تعريف المتغير up على أنه signed عشان لما يكون عندى صفر واقلله يصبح -1 أما لو المتغير كان char لما أقلل الصفر بمقار واحد سيصبح الرقم 255 يعنى مش أقل من الصفر فالسجمنت مش هتعرض 9 .

توصيل الدائرة :



المشروع الخامس

فكرة المشروع:

عمل برنامج يتكون من 4 مفاتيح عند الضغط على الاول يعد تصاعدى وعند الضغط على التانى يعد تنازلى وعند الضغط على الثالث يعد الاعداد الزوجية وعند الضغط على الرابع يعد الارقام الفردية .

الكود :

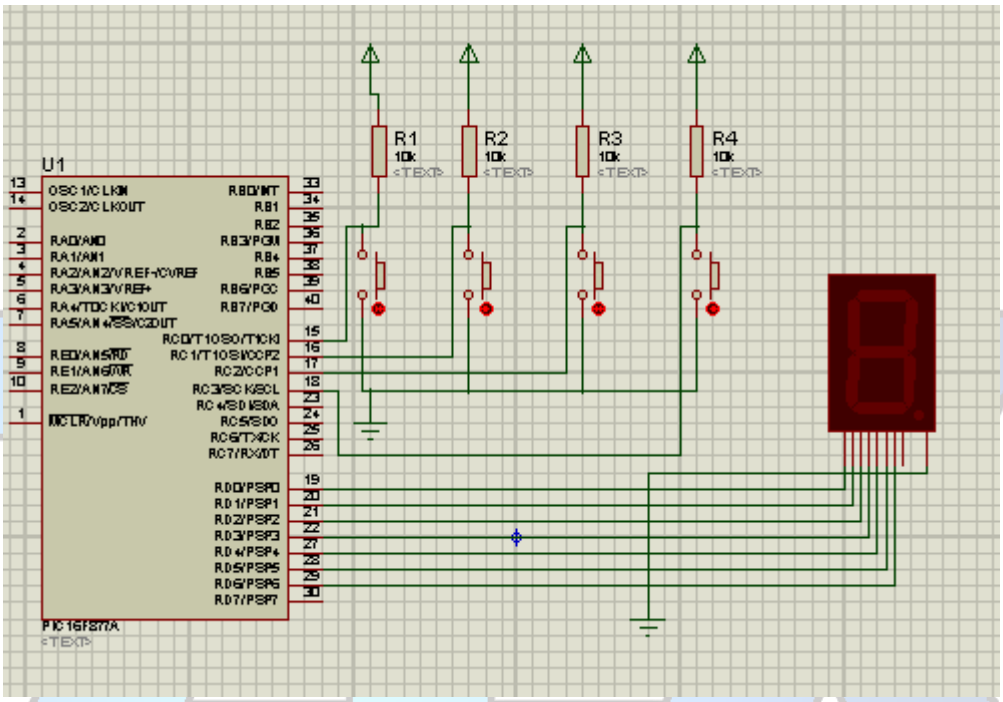
```
 unsigned short count (unsigned short num); // define function at the begining of the program
unsigned char up=0;
void main() {
trisd=0; //portd is output
trisc=255; //portc is input
portd=0; // put zero at portd as initial value
for (;;){ //infinite loop
if (rc0_bit==0){
delay_ms (10);
if (rc0_bit==0){

for (up=0;up<10;up++){   // UP COUNT 0:9
portd=count (up); //call function count
 delay_ms (1000);
 }
while (rc0_bit==0){}
delay_ms (10); // delay 1 sec
}
}
if (rc1_bit==0){  //DOWN COUNT 9:0
delay_ms (10);
if (rc1_bit==0){
for (up=9;up>=0;up--){
portd=count (up);
 delay_ms (1000);
 }
while (rc1_bit==0){}
delay_ms (10); // delay 1 sec

}
}
if (rc2_bit==0){  //EVEN COUNT 0,2,4,6,8
delay_ms (10);
if (rc2_bit==0){
for (up=0;up<=4;up++){
portd=count (up*2);
 delay_ms (1000);
 }
while (rc2_bit==0){}
delay_ms (10); // delay 1 sec
}
}
if (rc3_bit==0){     //ODD COUNT 1,3,5,7 ,9
delay_ms (10);
if (rc3_bit==0){
while (rc3_bit==0){}
delay_ms (10); // delay 1 sec
}
}
}
}
  unsigned short count (unsigned short num){
  switch (num){
  case 0 :return 0b00111111;
    case 1 :return 0b00000110;
      case 2 :return 0b01011011;
      case 3 :return 0b01001111;
         case 4 :return 0b01100110;
             case 5 :return 0b01101101;
               case 6:return 0b01111101;
          case 7 :return 0b00000111;
        case 8 :return 0b01111111;
         case 9 :return 0b01101111;
         }
         }
```

فى العداد الزوجى نجد أن الحلقة for مكونة من 4 مرات تكرار عندما تكون up=0 فانه من المعادلة up*2 نحصل على الرقم 0 ليعرض ع السجمنت وعند up=1 نحصل على 2 وعند up=3 نحصل على 6 وعند up=4 نحصل على 8 وكذلك فى العداد الفردى فان معادلته تكون (UP*2)+1 عندما UP=0 فانها تعرض 1 وهكذا .

توصيل الدائرة :



سؤال

هل معنى ذلك أننى إذا أردت أن استخدم السفن سيجمنت لا بد من شغل 8 بينات من الميكرو لصالح السفن سيجمنت ؟؟

وماذا لو كان لدى مخارج اخرى اكثر اهمية ومع ذلك فاننى احتاج السفن سيجمنت لعرض ارقام معينة فى مشروعى...هل اضحى بالسفن سيجمنت على حساب المخارج الاخرى أم العكس ؟

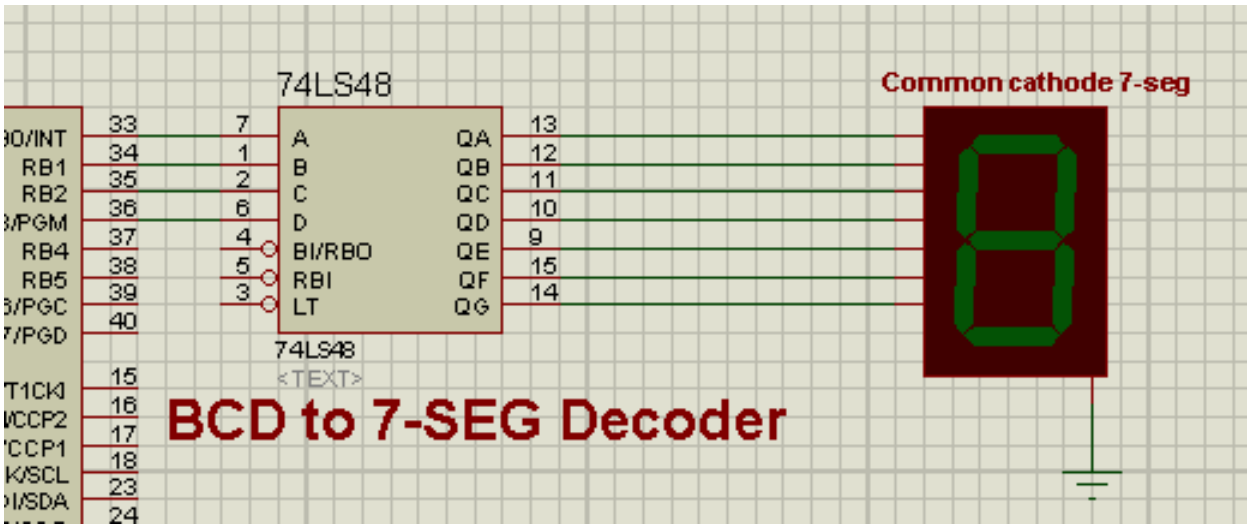
والجواب

انه يمكننى تشغيل السفن سيجمنت على 4 بينات فقط

من الميكرو (يعنى نصف عدد البينات) وذلك باستخدام دائرة متكاملة تسمى (74LS48)

وهى عبارة عن (BCD to 7 segment Decoder)

وهذه صورة لها



حيث نلاحظ ان لها 4 مداخل فقط من الميكرو ..و8 مخارج للسفن سيجمنت

كيفية عملها :

ان كلمة BCD تعنى binary coded decimal بمعنى اننى يمكننى عمل تشفير لكل رقم عشرى لكى يكون له رقم مناظر بالاعداد الثنائية (0,1) ..

فمثلاً.... كما فى المثال نلاحظ ان المتكاملة تم توصيلها على البورت b فإذا قمنا بكتابة الامر:
portb =3;

فمعنى ذلك ان الميكرو سيقوم بترجمته الى الاتى :

Portb = 0b00000011;

وهذا يعنى انه تم تحويله الى اعداد ثنائية....

عندها ..تقوم المتكاملة (اختصارا يمكن تسميتها 7448) باخراج ارقام مكافئة للعدد 3 لاظهارها على السفن سيجمنت (ستقوم بتشغيل الليدات الداخلية فى السفن سيجمنت المسماة (a,b,c,d,g

وبذلك يظهر لدينا الرقم 3 بكل سهولة..

وهذه صورة توضح جدول التحويلات الخاص بالمتكاملة(7448)

حيث توضح كل مدخل (A,B,C,D) للمتكاملة وما يكافئه من مخرج على السفن سيجمنت(a,b,c,d,e,f,g)

TRUTH TABLE
SN54/74LS48

	INPUTS							OUTPUTS							
DECIMAL OR FUNCTION	$\overline{LT}$	$\overline{RBI}$	D	C	B	A	$\overline{BI/RBO}$	a	b	c	d	e	f	g	NOTE
0	H	H	L	L	L	L	H	H	H	H	H	H	H	L	1
1	H	X	L	L	L	H	H	L	H	H	L	L	L	L	1
2	H	X	L	L	H	L	H	H	H	L	H	H	L	H	
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H	
4	H	X	L	H	L	L	H	L	H	H	L	L	H	H	
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H	
6	H	X	L	H	H	L	H	L	L	H	H	H	H	H	
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L	
8	H	X	H	L	L	L	H	H	H	H	H	H	H	H	
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H	
10	H	X	H	L	H	L	H	L	L	L	H	H	L	H	
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H	
12	H	X	H	H	L	L	H	L	H	L	L	L	H	H	
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H	
14	H	X	H	H	H	L	H	L	L	L	H	H	H	H	
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L	
BI	X	X	X	X	X	X	L	L	L	L	L	L	L	L	2
RBI	H	L	L	L	L	L	L	L	L	L	L	L	L	L	3
LT	L	X	X	X	X	X	H	H	H	H	H	H	H	H	4

وبذلك يتبين لنا مدى السهولة التى وفرتها لنا هذه المتكاملة الرائعة

وستجد فى الملفات المرفقة مشروع لعداد تصاعدى بهذه المتكاملة ويمكنك عمل باقى انواع العدادات تباعاً بكل يسر وسهولة.

ماذا تعني المصفوفات ؟

لكي نتعرف على معنى المصفوفات أخبرني كيف تستطيع حجز خمس متغيرات مثلا ؟ .. الإجابة : سأكتب الكود التالي :

```
char x1; char x2 ; char x3; char x4; char x5;
```

وبالطبع يمكن اختيار أسماء أخرى للمتغيرات.

أو يمكن كتباتها كما يلي :

```
char x1,x2,x3,x4,x5;
```

كلام جميل .. ولكن ماذا لو أردت أن تحجز خمسين متغير ، أو مئة متغير ؟ هل ستستخدم نفس الطريقة ؟

إذا استخدمنا نفس الطريقة فسيكون الأمر مرهق جداً . وهنا تأتي أهمية المصفوفات حيث أننا يمكننا حجز مئة متغير أو حتى ألف متغير في سطر واحد فقط ، فعلى سبيل المثال لو أردنا أن حجز ١٠٠ متغير كلهم من النوع char فيمكننا كتابة الكود التالي :

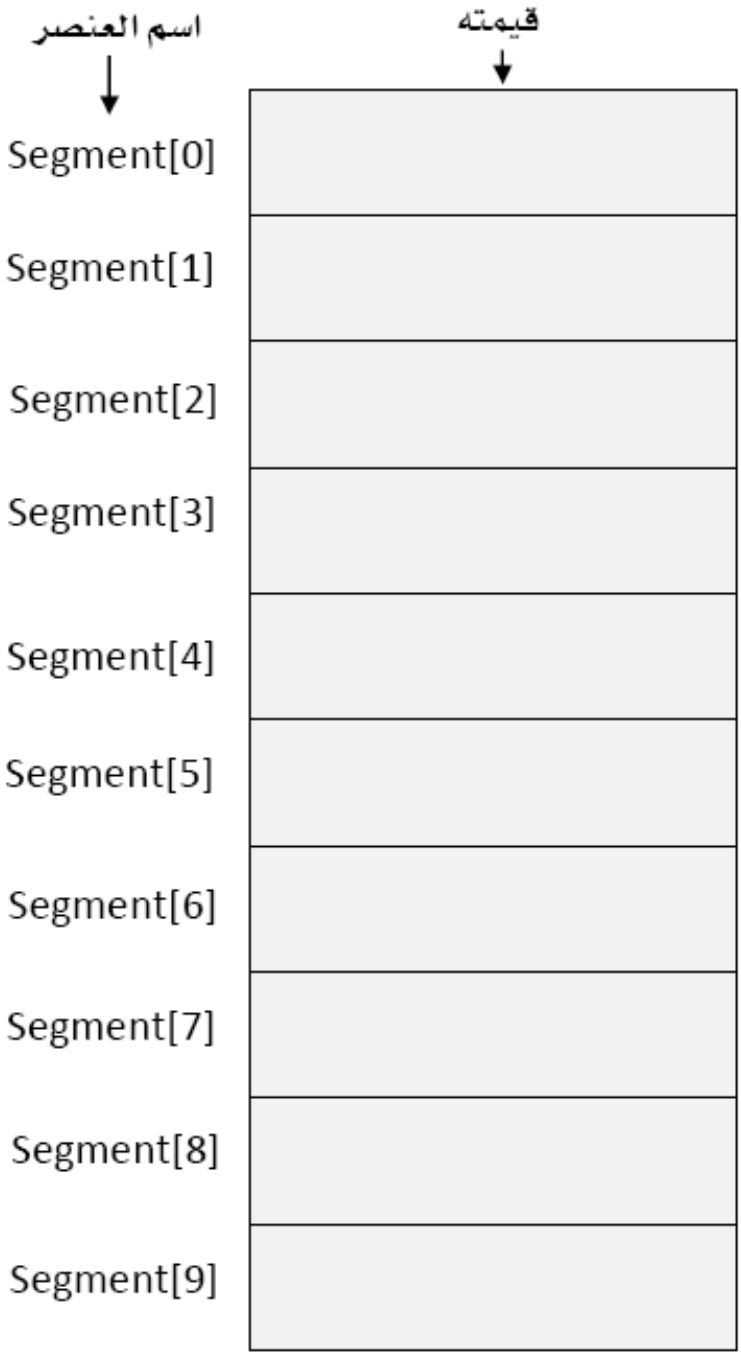
```
char ahmad[100];
```

بهذا الأمر تم عمل مصفوفة تتكون من ١٠٠ عنصر (١٠٠ متغير) وتم تسمية هذه المصفوفة بـ ahmad. إنهم الآن مئة متغير ، يبدأ ترقيمهم من الرقم صفر إلى ٩٩ . فالعنصر الأول اسمه ahmad[0] والعنصر الثاني اسمه ahmad[1] والعنصر الثالث هو ahmad[2] ... وهكذا .

ملحوظة : هناك فرق بين رقم العنصر وبين قيمة العنصر . فمثلا mm[4] هو اسم المتغير الخامس في المصفوفة التي أنشأناها وليس قيمته فليس شرطًا أن تكون قيمته تساوي أربعة .

عندما ننشئ مصفوفة بكتابتنا للأمر char segment[10]; فكأننا قمنا بعمل جدول أو مصفوفة كما

بالشكل التالي :



قيمة العنصر ربما تكون صفر أو واحد أو .. وهي قيمة محصورة بين الصفر و ٢٥٥ لأننا اخترنا النوع char

لو أردنا أن نجعل أول عنصر في المصفوفة يساوي خمسة فسنكتب الأمر segment[0]=5; ولو أردنا أن نجعل قيمة العنصر الأخير تساوي عشرة فسنكتب الأمر segment[9]=10; .. وهكذا .

معلومة :

يمكننا عمل مصفوفة مع تحديد قيمة كل عنصر في خطوة واحدة

مثال : char Ashraf[4]={ 5 ,10 ,100 ,20 };

والآن هيا بنا لنستخدم المصفوفات مع السفن سيجمنت ..!!

بسم الله وعلى بركة الله نبدأ .. سنستخدم سفن سيجمنت من النوع common cathode وبنفس التوصيلة التي استخدمناها في التجربة السابقة ، فلإظهار الرقم صفر نجعل portb يساوي 00111111 ولإظهار الرقم واحد نجعل portb يساوي 00000110 .. وهكذا

إذن ما رأيك أن نقوم بعمل مصفوفة اسمها segment بحيث يكون العنصر segment[0] يحتوي على شكل الرقم صفر أي 00111111 والعنصر segment[1] يحتوي على شكل الرقم واحد 00000110

والآن سنكتب كود يظهر الأرقام من صفر إلى تسعة على السفن سيجمنت .

```
void main( )
{
char segment [10]={0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,0b01101101,
0b011111101,0b00000111,0b01111111,0b01101111};
char x;
trisb=0;
while(1) {
        for(x=0;x<10;x++) {  portb=segment[x];
                             delay_ms(500);
                             }
       }
}
```

لعلك سألت نفسك وقلت لدينا الآن طريقتان لاستخدام السفن سيجمنت بشكل مباشر إما باستخدام جمل شرطية (جمل if أو switch.. case) وإما باستخدام المصفوفات ، فما الفرق الجوهري بينهما مع أن كلاهما يؤدي نفس الوظيفة ؟؟

لقد ذكرت في بداية الكتاب أن الميكروكنترولر يحتوي على ذاكرة من النوع RAM وذاكرة من النوع ROM وكذلك بروسيسور و وحدة إدخال وإخراج لكن إلى الآن لم نتطرق إلى تفاصيل في ذلك .

نحن الآن عندما نحجز متغير فإننا نحجزه في الـ RAM ويتميز هذا النوع من الذاكرة بأنه قابل للقراءة والكتابة وبسرعة عالية جداً .

ولكن ماذا عن الكود الذي نكتبه والذي يحدد وظيفة الميكروكنترولر ، أين يتم تخزينه ؟ هل هو أيضا يخزن في الـ RAM ؟ الإجابة .. لا بل يتم تخزينه في الـ ROM بحيث أننا نكتب البرنامج باستخدام لغة برمجة ثم عن طريق جهاز البرمجة (المبرمجة كما يسميها البعض) تتم الكتابة على الـ ROM . وعند تشغيل الدائرة التي تحتوي على الميكروكنترولر يقوم البروسيسور (المعالج) بتنفيذ الأوامر الموجودة في الـ ROM .

وفي الغالب يكون حجم الـ ROM أكبر بكثير من الـ RAM .

بعد هذه اللمحة السريعة دعنا نرجع إلى سؤالنا الأساسي لدينا الآن طريقتان لاستخدام السفن سيجمنت بشكل مباشر إما باستخدام جمل شرطية (جمل if أو switch.. case) وإما باستخدام المصفوفات ، فما الفرق الجوهري بينهما؟؟

الإجابة بكل بساطة أنه إذا استخدمنا الجمل الشرطية فإننا نأخذ مساحة أكبر في الـ ROM والمكسب هنا هو توفير الـ RAM لأن الجمل الشرطية سيتم تخزينها في الـ ROM . إما إذا استخدمنا المصفوفات أو المتغيرات فإننا بذلك نستخدم الـ RAM ونوفر في الـ ROM .

وللعلم ، فإن معظم التطبيقات البسيطة لن يؤثر فيها اختيارك لإحدى الطريقتين السابقتين ، ولكن أحياناً وفي بعض التطبيقات نجد أنه لا يوجد مكان في الـ RAM وعند ترجمة البرنامج تظهر رسالة تفيد بعدم وجود ذاكرة كافية (NO enough Ram) ، هنا في هذه الحالة نحاول بقدر الإمكان التوفير من الـ RAM وبالتالي لو كنا نستخدم سفن سيجمنت فسنستخدم الجمل الشرطية أو أي طريقة أخرى تجعل التخزين في الـ ROM وليس الـ RAM .

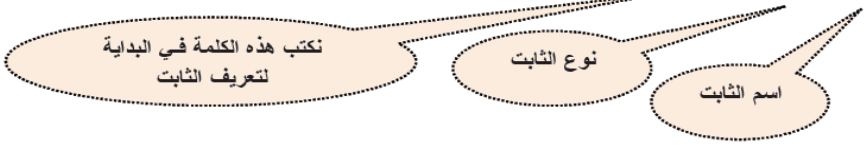
وبما أننا تحدثنا عن الذاكرة ، يجدر بنا الإشارة الآن إلى الثوابت .

فعلى سبيل المثال لو أردنا استخدام ثابت معين وليكن ٣.١٤ بحيث أنه سيتم استخدام

هذا الرقم في بعض العمليات(كحساب مساحة الدائرة أو محيطها مثلا) ولكن دون الحاجة إلى تغيير هذا الرقم .

ما سنفعله لإنشاء ثابت هو نفس الطريقة التي كنا نستخدمها مع المتغيرات ولكن نكتب كلمة const قبل النوع وإليك المثال التالي :

مثال : const float PI=3.14;



جميل نحن الآن أنشأنا ثابت اسمه PI وجعلنا قيمته تساوي ٣.١٤ ويجب أن نعلم أن هذه القيمة لا يمكن أن تتغير فلا يمكننا في أي وقت أن نكتب مثلا PI++ أو أي عملية تغير من قيمة PI .

من الممكن عمل مصفوفة أيضا كلها عبارة عن ثوابت وذلك أيضا بإضافة كلمة const كما ذكرنا .

وهنا تأتي الطريقة الثالثة في استخدام السفن سيجمنت فيمكننا كتابة الكود التالي :

```
void main( ){
const char segment [10]={0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,
0b01101101,0b011111101,0b00000111,0b01111111,0b01101111};
char x;   trisb=0;
while(1) {
        for(x=0;x<10;x++) {  portb=segment[x];
                              delay_ms(1000);
                            }
      }
}
```

الفرق الوحيد هو أننا حولنا المصفوفة من مصفوفة متغيرات إلى مصفوفة ثوابت ، وبذلك قمنا بعمل توفير كبير في استخدام الذاكرة العشوائية RAM.

لاحظ في لغة مايكروسي عند ترجمة المشروع ليتحول إلى ملف hex ظهور رسالة توضح ما تم استغلاله من RAM ، ROM والمساحة المتبقية من كل منهما .



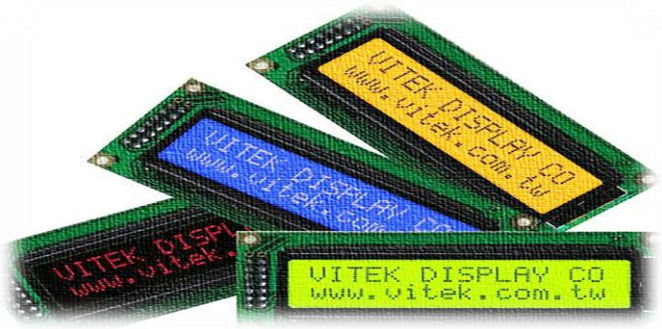
Chapter (6)

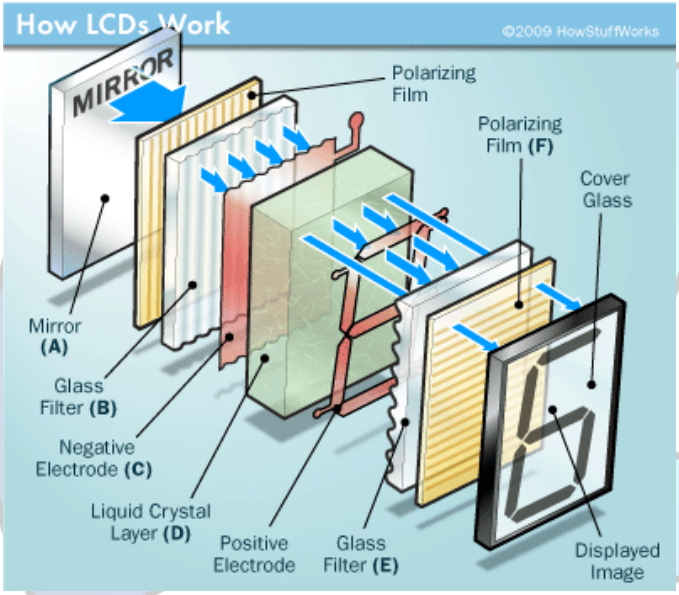
LCD



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

LCD





ما هى شاشة ال LCD ومما تتكون ؟

شاشة ال LCD هى عبارة عن شاشة تتكون من نقاط صغيرة من بلورات سائلة ويتم التحكم فيها من خلال التيار الكهربى وهذا يعنى اننا يمكن ان نتحكم فى كل نقطة على الشاشة.. وفى التلفزيونات الكبيرة هذا يعنى التحكم فى ملايين النقاط .

وبما اننا نتحكم فى كل نقطة اذن هذا يعنى انة يمكن صنع اشكال وانواع لا حصر لها وكلاً على حسب استخدامه
ولفظ LCD هو اختصار لجملة (Liquid Crystal Display) ومعناها بالعربية (عرض البلورات السائلة)

لماذا سميت بشاشة عرض البلورات السائلة؟

نعلم أن المواد في الطبيعة إما في الحالة الصلبة أو السائلة أو الغازية فالحالة الصلبة تكون فيها جزيئات المادة مرتبة باتجاه محدد وفي مواقع محددة بالنسبة لبعضها البعض أي لا تتحرك. أما في الحالة السائلة فإن جزيئاتها تكون في حالة حركة مستمرة ولا يجمعها اتجاه ترتيب محدد. ولكن هناك بعض المواد تكون في حالة وسطية أي بين السائل والصلب حيث تحافظ جزيئات المادة في هذه الحالة على اتجاه ترتيبها كما في جزيئات المادة الصلبة ولكن في نفس الوقت تتحرك مثل جزيئات الحالة السائلة، وهذا يعني أن البلورات السائلة هي ليست حالة صلبة وليست حالة سائلة ولكن بين الحالتين معا ومن هنا جاءت التسمية بالبلورات السائلة.

الفكرة الفيزيائية لعمل شاشات العرض التي تعتمد على البلورات السائلة

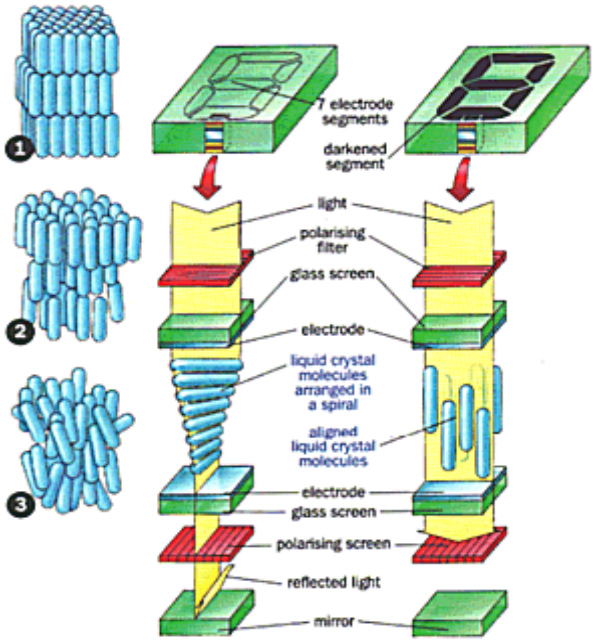
لتصنيع شاشة عرض من البلورات السائلة نستخدم لوحين من الزجاج المستقطب للضوء.. وهو عبارة عن مواد من البوليمر تحتوي على شرائح ميكروسكوبية (لا ترى بالعين المجردة)

تغطي احد سطحي لوح الزجاج الذي لا يحتوي على شريحة الاستقطاب. يتم ضبط الشرائح الميكروسكوبية لتكون في نفس اتجاه استقطاب الشريحة المثبتة على السطح المقابل. تتم بعد ذلك إضافة طبقة رقيقة من البلورات السائلة ذات الطور الدوار. تعمل طبقة الشرائح الميكروسكوبية على توجيه البلورات السائلة لتصطف في اتجاه تلك الشرائح. يتم وضع الطبقة الأخرى من الزجاج ولكن مع التأكد ان شريحة الاستقطاب عمودية على اتجاه استقطاب الشريحة الأولي. تترتب الطبقات المتعاقبة من البلورات السائلة ذات الطور الدوار الملتوي بعضها فوق بعض من بدوران تدريجي يصل إلى 90 درجة بالنسبة لترتيب الطبقة الأولى

عندما يسقط الضوء على الشريحة الزجاجية الأولي فإنها تعمل على استقطاب الضوء، ومن ثم تعمل جزيئات البلورات السائلة في كل طبقة على توجيه الضوء إلى الطبقة التي تليها مع تغير مستوى استقطاب الضوء. وعندما يصل الضوء للطبقة الأخيرة من طبقات البلورات السائلة فإنه يكون مستقطب في نفس اتجاء جزيئات تلك الطبقة وبالتالي ينفذ الضوء منها
وعند تطبيق مجال كهربي على جزيئات البلورات السائلة فإنها لا تلتوي وبالتالي فإن الضوء لا يمكن ان ينفذ من الجهة الأخرى

.

وهذه صورة توضح كيف تترتب جزيئات الكريستال السائلة فى حالة تشغيلها و اغلاقها



ما هى انواع شاشات ال LCD ؟

هناك العديد من انواع شاشات الLCD و هذه بعض أنواعها :

CHARACTER LCD

وهى تكون عبارة عن عدد من الصفوف والمربعات وكل مربع يطلق علية Character وكل مربع يكتب فية رمز او حرف او رقم واحد فقط أى اذا كانت صف واحد و16 مربع يمكننا عرض 16 رقم او 16 حرف او 16 رمز عليها مرة واحدة , وتقاس بمقدار الصفوف والمربعات بداخلها وهذه صورة لها



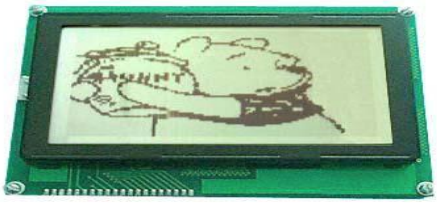
SEGMENT TYPE LCD

وهى ذات اشكال كثيرة جدا ولها اشكال ثابتة للتحكم بها لهذا تصنع على حسب الطلب ليس لها مقاييس محددة بل بالطلب وهذه صورة لها



GRAPHICAL LCD

وهى مثل Color LCD فى التركيب ولكن بدون الوان اى يمكن عرض اى شىء عليها ولكن بدقة اقل منColor LCD ويمكن الرسم عليها ايضا وتقاس بالبيكسل وهذه صورة لها



COLOR LCD

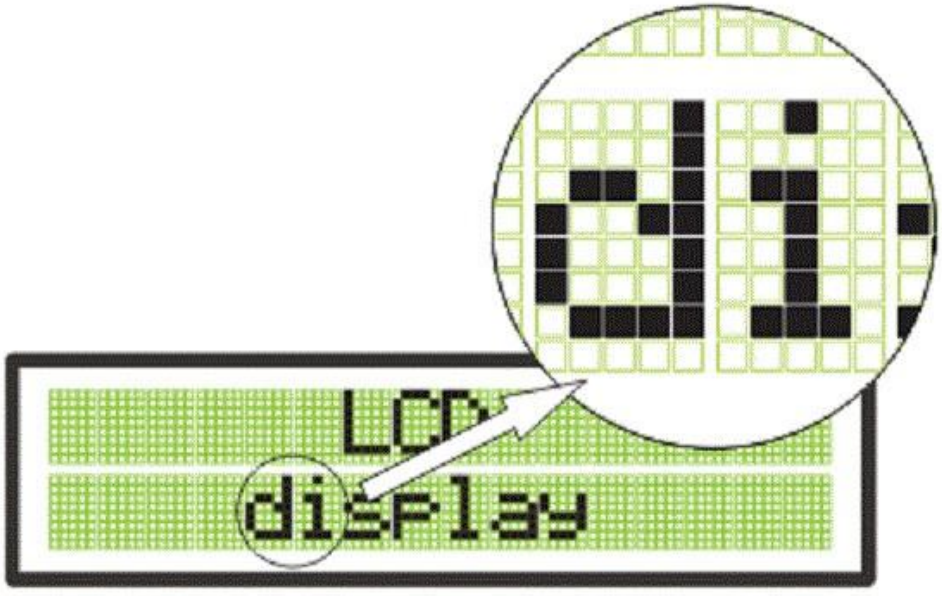
وهى مثل الموجودة فى التلفزيونات والموبيلات الملونة وتقاس بالبكسل وجودة الالوان وهذه صورة لها



ونظراًلأن اغلب مشاريع المبتدئين تكون بأول نوع وهو ال character LCD لذلك سنكتفى بشرح هذا النوع فقط

اولا : مم يتكون هذا النوع وكيف يعمل ؟

هذا النوع عبارة عن مربعات موضوعه على شكل اعمدة وصفوف وكل مربع مكون من عدد من البكسل اما 5*8 او 5*11 وهذه الارقام هى عدد البكسل لمربع واحد بس وهذه الصورة توضح هذا الكلام (5*8)



وهذه صورة اخرى تبين المربعات وازاى بتتشكل عليها الحروف والكلمات

وتقاس مساحة هذه الشاشة بعدد هذه المربعات وهذا ما يقال علية ان الشاشة مثلا 16*2 هذا يعنى ان هناك صفان فى كل صف 16 مربع اى الشاشة كاملة تحتوى على 32 مربع اى 32 حرف او رمز او رقم يمكن عرضهم مرة واحدة .

ولكن كيف سنتحكم فى كل هذا العدد من البكسل و لو الشاشة اكبر شوية دى تبقى حاجة معقدة جدا علينا وعايزة محترفين علشان يتعاملوا معاها.. الشركات المصنعة قامت بكل هذا بالنيابة عنك وقامت بوضع ICs خاصة للتعامل مع الشاشة والتحكم فيها فقط كل ما عليك انك هتتخاطب مع الشاشة ببروتوكول محدد .. ومن خلال البروتوكول دة (وهذا البروتوكول يسمى بالاسكى كود) ال IC الى فى الشاشة هيعرف انت بعت انهى حرف او انهى رمز او انهى رقم وهيكتبهولك على الشاشة وكل دة مخزن على الذاكرة بتاعته الداخلية مثلا لو انت بعت الاسكى كود بتاع الحرف" A " هو عارف هيشغل انهى بكسل و هيطفى انهى بكسل فى المربع الى انت اخترتة علشان يظهرلك الى انت بعتة وطبعا الطريقة دى بتسهل علينا شغل كتير اوى ودة شكل الشاشة وشكل ال IC الى بيقوم بالوظيفة دى .





لاحظ النقطة السوداء اللى على ظهر الشاشة هى دى الIC .
وبالتالى نفهم من كدة ان الكتابة على الشاشة دى محدودة نوعا ما لاننا لو كتبنا اى حاجة غير المتسجل على ذاكرة ال IC مش هيظهر حاجة لان الشاشة متعرفتش عليه وفعلا دة صحيح لو جبت شاشة من دول وكتبت عليها حرف عربى مش هيتعرض والشاشة هتخرف لان الشاشة متعرفش الشكل دة ودلوقتى بدئوا ينتجوا شاشات تقرا الحروف العربى وتظهرها

ثانياً: كيف نتعامل مع هذه الشاشات من خلال الميكروسى و ازاى نقدر نكتب عليها؟

هناك عدة اوامر يتم كتابتها على برنامج الميكروسى بحيث يمكن تشغيل هذه الشاشات والتعامل معها من خلال هذه الاوامر وهى اوامر بسيطة جداً وهى :

```
Lcd_Init
Lcd_Out
Lcd_Out_Cp
Lcd_Chr
Lcd_Chr_Cp
Lcd_Cmd
```

وسنقوم بشرحها امراً امراً :

اول امر هو :

```
Lcd_Init();
```

وهذا الامر عبارة عن داله التعريف اى الامر الذى يخبر الميكروكونترولر بان هناك شاشة سوف تركب علية وان هناك بروتوكول يجب اتباعه مع هذه الشاشة فى العمل .

ثانى امر هو :

```
Lcd_Out
```

هذا الامر يخرج البيانات على الشاشة بمكان انت تحددة يعنى لو عايز تخرج كلمة او عدد فى مكان محدد مثلا الصف الثانى وبداية اظهار هذه الكلمة تكون فى المربع الخامس هذا الامر هو ما يفعل ذلك لك .
وتوضع الكلمة داخل هذا الرمز "هنا تضع الكلمة" (double quotations) مهما كان حجم الكلمة ولكن اذا زادت عن عدد مربعات الشاشة فلن يظهر البقية ويمكنك ايضا كتابة كلمة داخل متغير ثم كتابة اسم المتغير داخل هذا الامر وهو سوف يعرض الكلمة التى داخل هذا المتغير
مثلا اذا اردنا اظهار كلمة Hello فى الصف الاول العمود السابع
تكون هكذا

```
Lcd_Out(1, 7, “Hello”);
```

هنا الرقم الاول هو رقم الصف وهنا هو 1 والرقم الثانى هو رقم المربع وهو هنا 7 ثم الكلمة بين هذا الرمز " " وهنا حرف ال H سيكون فى المربع السابع اى بداية الكلمة اما الحرف e فسيكون فى المربع الثامن وهكذا بالتتابع

ثالث امر هو :

```
Lcd_Out_Cp
```

الفرق بين هذا الامر والامر الذى قبله ان هذا الامر لا يخرج الكلمات او الاعداد حسب المكان الذى تريد بل حسب مكان اخر مربع قمت بالكتابة علية ويمكن القول حسب مكان المؤشر اى انك اذا كنت كتبت كلمة Go على المربع الاول فى الصف الثانى هذا يعنى ان المربع الاول تم كتابة حرف G علية ثم المربع الثانى تم كتابة الحرف O علية
اذن المؤشر يقف على المربع الثانى وهو اخر ما تم الكتابة علية اذن قم بكتابة الكلمة Mohamedعلى الشاشة من خلال هذا الامر الان اى هكذا

```
Lcd_Out_Cp("Mohamed");
```

كما ترى بدون تحديد اماكن الصف او المربع ..اين سوف يبدء اى اين سوف يكون حرف ال M وهو الحرف الاول من الكلمة ؟
يكون بعد اخر مربع تم الكتابة علية او بعد المربع الذى يوجد علية المؤشر وهنا هو المربع الثالث اى الشاشة سوف تظهر الكلمات كلها هكذا GoMohamed
ويمكنك وضع فاصل قبل كلمة Mohamed هكذا داخل الرموز"Mohamed " ليظهر فاصل بين الكلمتين كما ترى الفراغ الصغير بجانب الكلمة

الامر الرابع هو :

```
Lcd_Chr
```

هذا الامر يقوم باظهار حرف واحد فقط او رقم واحد فقط فى مكان تحددة انت واذا كتبت علية اكثر من ذلك فى المرة الواحدة فلن يظهر شيئ ويجب وضع الحرف او الرقم داخل هذا الرمز ' ' هنا تضع الحرف او الرقم (single quotation) وهو الافضل فى اظهار رقم او حرف واحد فى المرة الواحدة
واذا اردنا اظهار الحرف R فى الصف الاول المربع الخامس يكون هكذا

```
Lcd_Chr(1, 5, 'R');
```

فالصف هو الرقم الاول وهو 1 اما المربع فالرقم الثانى وهو 5 ثم الحرف بين هذا الرمز ' '

الامر الخامس هو :

```
Lcd_Chr_Cp
```

وهو لاظهار رقم او حرف واحد فقط فى كل مرة واذا كتبت علية اكثر من ذلك فى المرة الواحدة فلن يظهر شى ويجب وضع الحرف او الرقم داخل هذا الرمز 'هنا تضع الحرف او الرقم ' وهو الادق لاظهار حرف او رقم فى المرة الواحدة ولكن انت لا تحدد مكان العرض بل بمكان المؤشر او اخر مربع تم الكتابه علية واذا اردت كتابة الحرف W على اخر مكان للمؤشر يكون هكذا

```
Lcd_Chr_Cp('W');
```

الامر السادس هو:

```
Lcd_Cmd
```

هذا الامر هو لكتابة اشكال واعطاء اوامر للشاشة بمعنى انه اذا اردت مسح الشاشة او تحريك المؤشر او تحرك الكلمات على الشاشة او او او الى اخر الاوامر..

وهذه الصورة بها الاوامر التى تستخدم للشاشة :

Available Lcd Commands

Lcd Command	Purpose
_LCD_FIRST_ROW	Move cursor to the 1st row
_LCD_SECOND_ROW	Move cursor to the 2nd row
_LCD_THIRD_ROW	Move cursor to the 3rd row
_LCD_FOURTH_ROW	Move cursor to the 4th row
_LCD_CLEAR	Clear display
_LCD_RETURN_HOME	Return cursor to home position, returns a shifted display to its original position. Display data RAM is unaffected.
_LCD_CURSOR_OFF	Turn off cursor
_LCD_UNDERLINE_ON	Underline cursor on
_LCD_BLINK_CURSOR_ON	Blink cursor on
_LCD_MOVE_CURSOR_LEFT	Move cursor left without changing display data RAM
_LCD_MOVE_CURSOR_RIGHT	Move cursor right without changing display data RAM
_LCD_TURN_ON	Turn Lcd display on
_LCD_TURN_OFF	Turn Lcd display off
_LCD_SHIFT_LEFT	Shift display left without changing display data RAM
_LCD_SHIFT_RIGHT	Shift display right without changing display data RAM

فمثلاً لكتابة امر مسح الشاشة نكتب هكذا

```
Lcd_Cmd(_LCD_CLEAR);
```

ولكتابة امر قفل المؤشر وعدم اظهارة يكتب هكذا

```
Lcd_Cmd(_LCD_CURSOR_OFF);
```

كيف يتم عرض أرقام على الشاشة:

إذا أردنا عرض رقم على الLCD ، نقوم بتحويل الرقم من int إلى string أولا حيث أن الLCD لا تستطيع عرض إلا string فقط

وللقيام بهذا نكتب الأمر التالي

```
int temp;                    //→1
char temp_txt[7];            //→2
inttostring(temp,temp_txt);  //→3
lcd_out(1,1,temp_txt);       //→4
```

سنقوم الآن بشرح كل سطر من الأسطر السابقة:

1. تم تعريف متغير من نوع integer وتم تسميته temp.
2. تم تعريف مصفوفة من الحروف(لأن لغة ال C ليس بها string) سعتها 7 وتم تسميتها temp_txt(إذا كنت ستقوم بتخزين int مثل هذه الحالة فيجب أن يكون طول المصفوفة هو7، ولكن إذا كنت تريد عرض float فيجب أن يكون طول المصفوفة هو 16، ويجب أن يكون الأمر التالي floattostr بدلا من inttostr).
3. تم نسخ الأرقام من المتغير temp إلى المصفوفة temp_txt.
4. سيتم عرض القيمة المخزنة في المصفوفة temp_txt على الLCD بداية من السطر الأول والحرف الأول.

إذا طبقنا هذا المثال سنلاحظ أن الرقم حتى وإن كان صغيرا ، يستغل مساحة من الLCD بمقدار 7 حروف(مساحة المصفوفة)

وللتغلب على هذه المشكلة نقوم بكتابة الآتي:
باعتبار أن الرقم المراد عرضه يتكون من ثلاث أرقام، وقد تم تخزينه في متغير يسمى temp

```
int temp, ons, tens, hun;  //→1
ons=temp%10;               //→2
temp=temp/10;              //→3
tens=temp%10;              //→4
hun=temp/10;               //→5
lcd_chr(1,1,hun+48);       //→6
lcd_chr_cp(tens+48);       //→7
lcd_chr_cp(ons+48);        //→8
```

سنقوم الآن بشرح كل سطر من الأسطر السابقة:

1. تم تعريف ثلاث متغيرات ons و tens و hun لأن الرقم المراد عرضه على الشاشة يتكون من ثلاث أرقام.
2. سيتم تخزين باقي قسمة الرقم temp على 10 في المتغير ons، وبما أن ons هو من نوع int، فسيتم في هذا السطر تخزين رقم الأحاد من الرقم temp فقط في المتغير ons
فعلى سبيل المثال إذا كانت قيمة المتغير temp هي 452 فعند قسمته على 10 سيكون الناتج 45 و باقي القسمة 2، وهذا هو رقم الأحاد من الرقم 452.
3. تم قسمة ال temp على 10 ثم تخزينه في temp مرة أخرى ، وبما أن temp هي من نوع int فسيتحول قيمة الtemp من 452 إلى 45.
4. كما تم في السطر الثاني سيتم أيضا في السطر الرابع، حيث سيتم تخزين رقم العشرات من الtemp في المتغير tens.
5. سيتم تخزين رقم المئات من الtemp في المتغير hun.
6. تم عرض قيمة رقم المئات في أول سطر في الشاشة ، في أول حرف.
وتم إضافة 48 إلى قيمة المئات لأن أمرlcd_chr يعرض الأسكي كود بتاع الرقم المكتوب، والأسكي كود بتاع رقم 0 هو 48، فتم إضافة 48 إلى أي رقم سيكون في المئات ليتكون الأسكي كود الصحيح المقابل للرقم.
7. سيتم فعل ما تم بالنسبة للمئات مع رقم العشرات.
8. سيتم فعل نا تم بالنسبة للمئات والعشرات مع الأحاد.

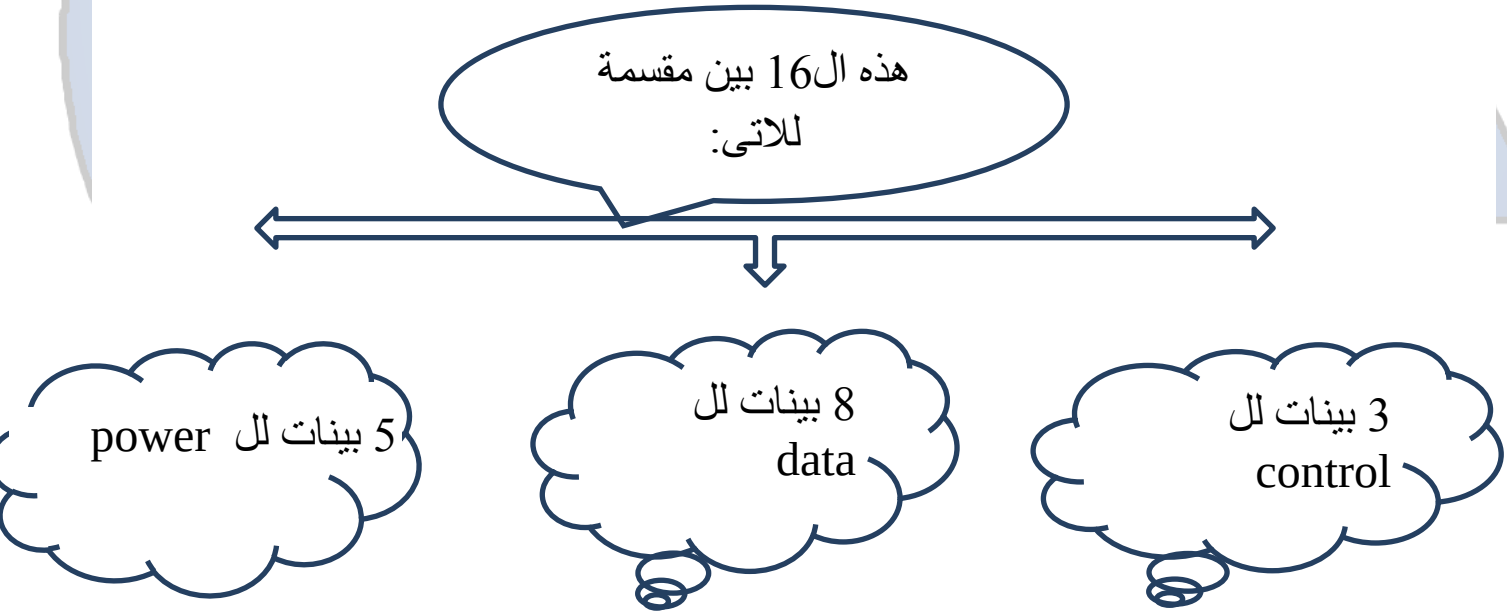
وبذلك تم عرض الرقم المكون من ثلاث أرقام في ثلاث خانات فقط، على عكس ما تم في الطريقة الأولى من حجز 7 أماكن للرقم.

ثالثاً: كيف يتم توصيل الشاشة عملياً وما الذى يستلزم كتابته فى البرنامج ميكروسى لهذا التوصيل ؟

لو نظرنا جيداً للشاشة سنجد ان لها **16 pin**



ولاحظ ان الرجل رقم 1 على أقصى اليسار والرجل رقم 16 على اقصى اليمين



وسنشرح ال16 بين بالتفصيل حسب ترتيبهم وسنضع بجانب كل بين حرف لنعرف هل هى لل power (P)او لل data (D) او لل control (C)

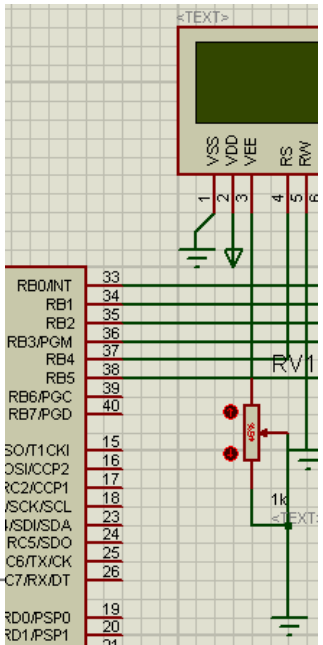
رقم البين	وظيفتها	نوعها
1	تغذى الشاشة بالجهد اللازم للتشغيل ويتم توصيلها عالارضى(الارضى مشترك للبيك ولكل الملحقات من شاشة وخلافه)	P
2	تغذى الشاشة بالجهد اللازم للتشغيل ويتم توصيلها عالموجب(5 فولت وهو موجب دائرة البيك)	P
3	مسئولة عن التباين* للشاشة ويتم توصيلها بالارضى للحصول على اعلى تباين (انظر معنى التباين بالاسفل)	P
4	مسئولة عن تحديد نوع الداتا الداخلة للشاشة هل هى معلومات يتم عرضها (high volt) ام اوامر يتم تنفيذها(low volt)[يتم تحديد ذلك من خلال البيك لذلك نقوم بتوصيلها على البين B4 فى البيك	C
5	مسئولة عن تحديد هل سيتم الكتابة على الشاشة (low volt) ام قراءة ما عليها (high volt) (يتم توصيلها عالارضى للكتابة عليها)	C
6	مسئولة عن اعطاء نبضات للشاشة من الميكروكونترولرلادخال الداتا اليها[يتم توصيلها بالبين B5 فى البيك	C
7	مسئولة عن تلقى الداتا القادمة من الميكروكونترولر(وهم ثمان ارجل من 7 الى 14) [لا يتم توصيلها]	D
8	[لا يتم توصيلها]	D
9	[يتم توصيلها]	D
10	[لا يتم توصيلها]	D
11	[يتم توصيلها بالبين B0 فى البيك]	D
12	[يتم توصيلها بالبين B1 فى البيك]	D
13	[يتم توصيلها بالبين B2 فى البيك]	D
14	[يتم توصيلها بالبين B3 فى البيك]	D
15	مسئولة عن شدة الاضاءة ويتم توصيلها بموجب الدائرة(5 فولت)	P
16	مسئولة عن شدة الاضاءة ويتم توصيلها بارضى الدائرة حيث ان الشاشة تحتوى على ليدات خلفية يتم تشغيلها عندما توصل الرجل 15 بالموجب وتوصل الرجل 16 بالارضى مما يساعد على رؤية ما يتم كتابته على الشاشة فى الظلام	P

***التباين** contrast في البصريات هو الاختلاف في اللون أو الاختلاف في الظل الذي يجعل الأشياء واضحة في الصورة ويفرقها عن بعضها . إذا انخفض معدل التباين في صورة صعب التفرقة بين الأشياء في الصورة ، ونقول أن الصورة ضبابية أو تعتريها غمامة.

ملاحظات :

أولا :

الشاشة الخضراء توصل علطول بالارضى لكن الشاشات الاخرى احيانا تكون غامقة او فاتحه فبوصل مقاومة متغيرة على الرجل 3 تكون فى حدود من 5 الى 10 كيلو اضبط بيها لون الشاشة لو وصلت الرجل 3 على الارضى وكان اللون واضح مش شرط اوصل المقاومة المتغيرة لكن لا أدع pin3 تكون float .



ثانيا :
الارجل 7,8,9,10 غير موصلين لانى بشتغل على 4 bit mode بنقل الداتا 4 bit ب4 bit وحيث أن ال char بيكون 4 bit فبيتم ارسال الداتا على مرتين دون الشعور بالفرق لان سرعة الميكرو كبيرة أما لو بشتغل على ال 8 bit mode باستخدم ال data bus كلها لذا فانه فى ال 4 bit mode فان 7,8,9,10 pin بتكون leave open وال 11,12,13,14 pin بتكون in use .

ملحوظة

فى الشاشات الالفا نوميريكال (زى بتاعتنا يعنى) نقوم بتوصيل 4 رجول للداتا فقط اما فى شاشات الكلر والجرافيك لا بد من توصيل ال8 ارجل.

ملحوظة :

التوصيلات عالبيك المذكورة فى الجدول بناءً على اوامر جاهزة يتم كتابتها من مكتبة برنامج الميكروسى وفى حالة الرغبة فى تغيير هذه الاماكن يتم تغييرها تباعاً فى الكود بالمكان الجديد بمعنى

ان هذه الاوامر ثابتة وما يتغير فيها هو الارجل التى سوف تربط بها البك بالشاشة وهى

```
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;
```

هنا ستجد ان كل طرف من الشاشة مكتوب بجواره اسم الرجل التى سوف يربط معها من الميكرو مثلا

```
LCD_RS at RB4_bit
```

هنا طرف الشاشة وهو LCD_RS سوف يربط مع الطرف RB4 اى الطرف رقم 4 فى البورت B فى الميكرو وهكذا واذا اردت مثلا تغير مكان تركيب الاطراف فغير فى الجزء الخاص بالبورت مثلا مكان RB4 يكون RA4 وهكذا وهذه الاوامر يجب كتابتها فى بداية الكود اى قبل ال

```
void main()
```

وداخله.

اسم ال lcd على البروتس هو LM016L display

المشروع الاول

برنامج لاظهار رسالة على الشاشة ((راجع الملفات المرفقة)).

المشروع الثانى

برنامج لاظهار رسالة متحركة ع الشاشة

```
        for (i=1;i<=16;i++){
Lcd_Out(1, i, "computer");
                    Lcd_Out(2, i, "engineering");
delay_ms (500);
    Lcd_Cmd(_LCD_CLEAR);
            delay_ms (30);
            }
            }
```

الفكرة انى بعرف متغير i بدخلة فى for loop لان عرض السطر 16 حرف فى الشاشة بجعل هذا المتغير هو رقم المربع اللى المفروض الكلمة تبدأ تتعرض من عنده وكل ما يزيد المتغير ده الكلمة تنتقل من المربع للمربع الذى يليه وبكده تظهر الكلمة كأنها تتحرك على الشاشة .

لاحظ بعرض الكلمة لمدة نصف ثانية لكن لماذا وضعنا أمر مسح الشاشة بعد كل عدة وذلك لانها سيعرض كلمة computer حرف ال c هيكون فى المربع الاول من الشاشة ثم تزداد i بمقدار 1 فسيعرض كلمة computer ولكن من اول المربع الثانى فيظهر من كلمة computer الاولى والتى كانت عند i=1 حرف ال c وتعرض كلمة computer الثانية والتى كانت عند i=2 ولا يظهر منها ايضا غير حرف ال c عند عرض كلمة computer والتى تكون عند i=3 وهكذا فيظهر ع الشاشة ccccccccomputer لهذا نضع أمر مسح الشاشة .

((راجع الملفات المرفقة)) .

المشروع الثالث

ال lcd مش بتعرض أرقام مباشرة ولكنها خاصة بال char وال string لذا بعلن عن متغير char ممكن يكون 16 حرف عن طريق الامر char txt [16]; وعندى متغير i من النوع int لذا احوله الى string عن طريق الامر IntToStr(i, txt); وبعد كده اعرض ال string وهذا مثال بسيط .

Example	
	int j = -4220; char txt[7]; ... IntToStr(j, txt); // txt is " -4220" (one blank here)

فكرة البرنامج :

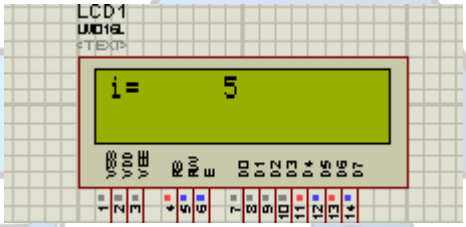
عرض أرقام من 0 الى 8 على الشاشة

```
     char str [16];
void main() {
int i;
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);
  for (;;){
  for (i=0;i<9;i++){
                Lcd_Cmd(_LCD_CLEAR);
     Lcd_Out(1, 1, "i=");
        IntToStr(i, str);
         Lcd_Out_Cp(str);
     delay_ms (500);

}
                    }

}
```

من المعلوم ان الدالة lcd_out_cp تظهر الرقم بعد علامة = مباشرة لكن المشكلة فى الامر inttostr أنه يغير شكل الداتا ويخلى الرقم يظهر على مسافة من = كما هو موضح بالشكل .



((راجع البرامج فى الملفات الملحقة))

المشروع الرابع

عمل عداد تصاعدى وعداد تنازى باستخدام السوتشات

((راجع البرامج فى الملفات الملحقة))

Chapter (7)

DC Motor



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013





بسم الله الرحمن الرحيم.....
إن شاء الله تعالى سنتعلم في هذا الدرس كيفية التحكم في DC Motors من حيث الإتجاه والسرعة عن طريق الميكرو كنترولر ، وكيفية إستخدام هذا النوع من المواتير في التطبيقات المختلفة.

سنتناول في هذا الدرس إن شاء الله:
1- مميزات محركات التيار المستمر.
2- نظرية تشغيل الموتور.
3- طرق معرفة مواصفات الموتور المجهولة.
4- دائرة تشغيل الموتور.
5- التحكم في إتجاه الموتور باستخدام H-Bridge.
6- التحكم في إتجاه الموتور باستخدام IC جاهزة.
7-التحكم في سرعة الموتور من خلال PWM.

***أولا: مميزات محركات التيار المستمر:**
يتميز هذا النوع من المحركات عن باقى المحركات بالاتى:
1- التكلفة المنخفضة.
2- توفرها.
3- سهولة التحكم.

ويعاب عليها عدم الدقة.

***ثانيا: نظرية تشغيل الموتور:**
تعتمد محركات التيار المستمر على الكهرومغناطيسية فهى تقوم بتحويل
القدرة الكهربائية الى قدرة ميكانيكية (حركية) (استخدم google اذا اردت معرفة نظرية العمل بالتفاصيل هناك شرح كثير بالصور يسهل فهم النظرية لم احب ان انقلها هنا حتى لا يتشعب

الموضوع .(غالب هذه المحركات يكون دخلها عبارة قطبين موجب وسالب يعتمد اتجاه حركة الموتور على على قطبية التوصيل.

المعادلات:

العلاقة بين الجهد الذى يغذى الموتور وسرعته علاقة طردية مباشرة

فكلما زاد الجهد زادت السرعة والعكس صحيح . اى ان

Speed=Kv*V.....(1)

Kv:::ثابت

V::::: جهد الدائرة المتكاملة التى تتحكم فى الترانزسنور (غالبا 5 فولت.(لكن هذا القانون غير دقيق لان الموتور يحتوى على مقاومة داخليةR هذه المقاومة تتسبب فى فقد فى الجهد مقداره (I*R) حسب قانون اوم

اذن القانون الامثل هو

Speed=Kv(V-I*R).....(1)

اذن اذا كان لدينا موتور بالمواصفات التالية

Rated 12 V

Rated 120 Rpm

فاذا غذينا الموتور بجهد 12 فولت فالسرعة لن تكون 120 بل اقل من ذلك بسبب الفقد فى الجهد.

ايضا العلاقة بين العزم (Torque) (القوة التى تنتج دوران الموتور) والتيار المسحوب علاقة طرية ايضا بمعنى كلما تتطلب الامر عزما كبيرا زاد التيار المسحوب والعكس صحيح اى

Torque=Kt*I(2)

Kt::: ثابت

I ::: التيار المسحوب من المصدر

ولكن ليس كل التيار المسحوب من المصدر سيتسبب فى هذا العزم

هناك فقد فى القدرة سببه تيار اخر هو Io تيار اللاحمل هو التيار المسحوب عندما لا يكون هناك عزم على الموتور او بمعنى آخر لا يوجد حمل.

وبنفس الطريقة القانون الامثل هو

Torque=Kt(I-Io)(2)

سنحتاج الى هاتين المعادلتين اذا اردنا ان نتحكم فى الموتور.

***ثالثا: طرق معرفة مواصفات الموتور المجهولة:**

طبعا افضل واسهل طريقة لمعرفة هذه المواصفات هى النظر فى ال"Data sheet" لكن هذا لا يكون متوفر دائما. فما الحل.

اولا يجب علينا ان نحدد مالذى يجب ان نبحث عنه:

1).....Speed=Kv(V-I*R)

2).....Torque=Kt(I-Io)

من المعادلتين السابقتين فان نحتاج لمعرفة الثوابت التالية

Kv,Kt,R,Io

وسنذكر هنا تفاصيل ايجاد هذه الثوابت:

1- المقاومة الداخلية للموتور R :

لا يمكن قياسها مباشرة باستخدام الأفوميتر الامر يحتاج الى بعض الترتيبات.

ا- قم بكبح العمود الدوار(الشافت) الغرض من هذا ضمان عدم تبدد جزء من القدرة الداخلة فى شكل قدرة ميكانيكية.

ب- ثم بتسليط جهد قليل جدا وكلما قل الجهد كان افضل ولا تقم بتسليط ال rated voltage فان ذلك سيؤدى الى تلف المحرك بسبب مرو تيار كبير جدا فى الملفات ثم قم بقياس التيار المار خلال المحرك (الموتور) . وبقسمة الجهد الذى قمت بادخاله على التيار تكون قد تحصلت على المقاومة R.

2- تيار اللاحمل: Io

أ- اترك عمود الادارة حرا من اى احمال.

ب- ستحتاج الى مصدر جهد متغير قم بتسليط جهد مناسب اقل من جهد الRated وابدأ فى زيادة الجهد تدريجيا مع ملاحظة قيمة التيار الداخل الى الموتور عند جهد معين ستجد ان اى زيادة فى الجهد لا تؤثر فى قيمة التيار والتى تكون ثابتة وهذه القيمة هى قيمة تيار اللاحمل.

اذا لم تكن تمتلك مصدر جهد متغير قم بتغذية الموتور بالجهد الذى ستسخدمه اثناء التطبيق . التيار المار هو تيار الاحمل.

3-ثابت السرعة: Kv

قم بتسليط الجهد ال Rated على الموتور ثم قم بقياس السرعة وبقسمة السرعة على الجهد تكون قد تحصلت على ثابت السرعة. بسيطة اليس كذلك ، لكن المشكلة تكمن فى قياس السرعة

طبعا هناك اجهزة مخصصة لهذا الغرض لكن التكلفة + عدم التوفر تمنع من استخدامها

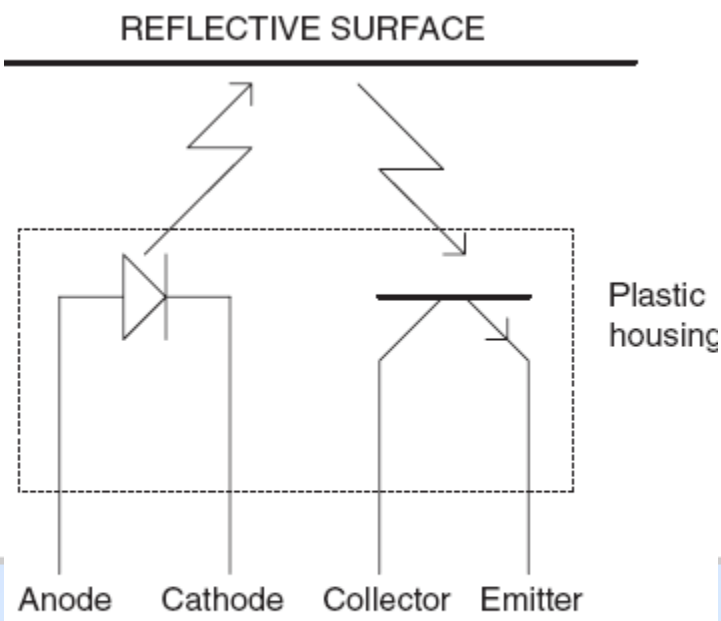
ويمكن انجاز هذا الامر بالطريقة التالية

1- طريقة استخدام الphoto interrupter sensor

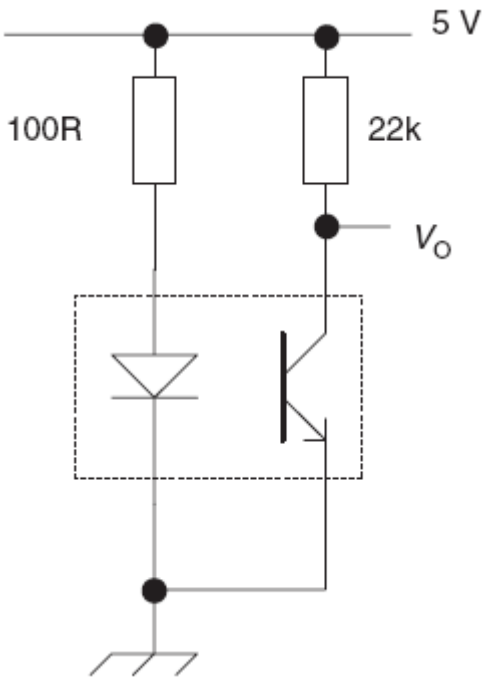
ويتم استخدام photo sensor كالذى فى الشكل التالى



وهو كما هو معلوم يتكون من مرسل و مستقبل



كما فى الشكل يعتمد خرج الدائرة على السطح الذى سيستقبل الاشارة فاذا كان السطح داكن فانه سيمتص الاشارة والعكس صحيح.
اذا استخدمنا الدائرة التالية



فان الخرج Vo سيكون 5 فولت فى حالة اللون الاسود و 0 فولت فى حالة اللون الابيض فاذا قمنا بربط ديسك قرص(يجب ان يون خفيف الوزن) ووضع السنسور كما فى الشكل التالى



فاذا قمنا بتشغيل الموتور فان الخرج سيكون على شكل سلسلة من النبضات فاذا افترضنا ان لدينا 16 زوجا من اللونين الابيض والاسود فهذا يعنى ان كل 16 نبضة تمثل دورة كاملة . فاذا قمنا بتشغيل الموتور لمدة دقيقة كاملة وبقياس عدد النبضات وبقسمتها على 16 نكون قد تعرفنا على سرعة الموتور.

ملحوظة: قياس النبضات سيكون باستخدام الميكرزكنترولر

(وسيكون برنامج قياس السرعة في المرفقات بإسم Motor speed test)

4-ايجاد ثابت العزمKt

النسبة بين ثابت العزم وثابت السرعة هى 1352

اى لايجاد قيمة ثابت العزم:

Kt=Kv/1352

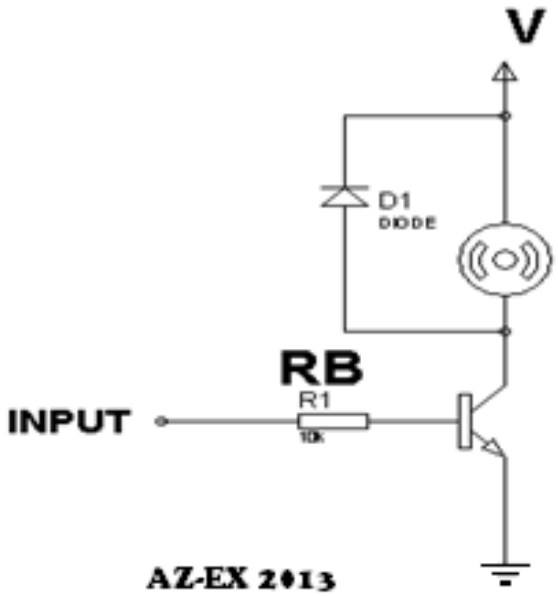
وبذلك نكون قد تمكنا من ايجاد كل الثوابت.

***رابعا: دائرة تشغيل الموتور:**

يمكن تشغيل الموتر بتوصيله بمصدر جهد مستمر بطارية مثلا.

لكن سنهتم بتوصيله بالدوائر الرقمية مثل الميكروكنترولر لان معظم الدوائر الرقمية لا تستطيع توفير تيار كافى لتشغيل الموتور فلا يمكن ربط الموتور مباشرة . لابد من توصيل الموتور بمصدر منفصل والتحكم فيه عن طريق مفتاح وهو هنا الترانزستور.

وهذه الدائرة تفى بالغرض



ما فائدة الدايود؟

لان الموتور هو من الاحمال الحثية وهى بطبيعتها تمنع التغير السريع فى التيار وهذا يحدث عند غلق المفتاح (جهد 0 على القاعدة) فان هذا الاصرار يجعل الملف يولد جهد عالى يكفى لاستمرار مرور التيار ويكون هذا الجهد معكوسا وسيمر التيار من خلال الدايود، لحماية الترانزيستور من مرور هذا التيار فيه فينهار.

مواصفات الترانزستور:

يجب ان تتوفر فى الترانزستور المواصفات التالية:

اولا

Hfe(min)>(5*I)/25

:::: Iتمثل التيار المسحوب بmA

ثانيا

Ic (max)>I

اى ان اقصى تيار للمجمع يجب ان يكون اكبر من التيار المسحوب بواسطة الموتور.

ثالثا

من نوع.NPN

رابعا

.RB=(V*hfe)/(5*I)

::: I *تمثل التيار المسحوب بmA

::::: Vجهد تغذية الدائرة المتكاملة (غالبا 5 فولت).

:::: RBمقاومة القاعدة بالكيلو اوم.

مواصفات الدايود:
1- جهد اعلى من ضعف جهد المصدر.
2- تيار اعلى من اقصى تيار متوقع.
وتضاف خاصية اخرى اذا اردنا ان نتحكم فى سرعة الموتور باستخدام تقنية ال PWM وهى ان يكون:
3- من نوع fast recovery Diode
trr<200nsec

***خامسا: التحكم في إتجاه الموتور :**

في إتجاه واحد One direction:

هنا فقط يتم تشغيل وايقاف الموتور وذلك باستخدام الدائرة السابقة

واذا افترضنا اننا قمنا بتوصيل قاعدة الترانزستور مع PORTB.B0 اذا لتشغيل الموتور نستخدم الامر التالى

كود:

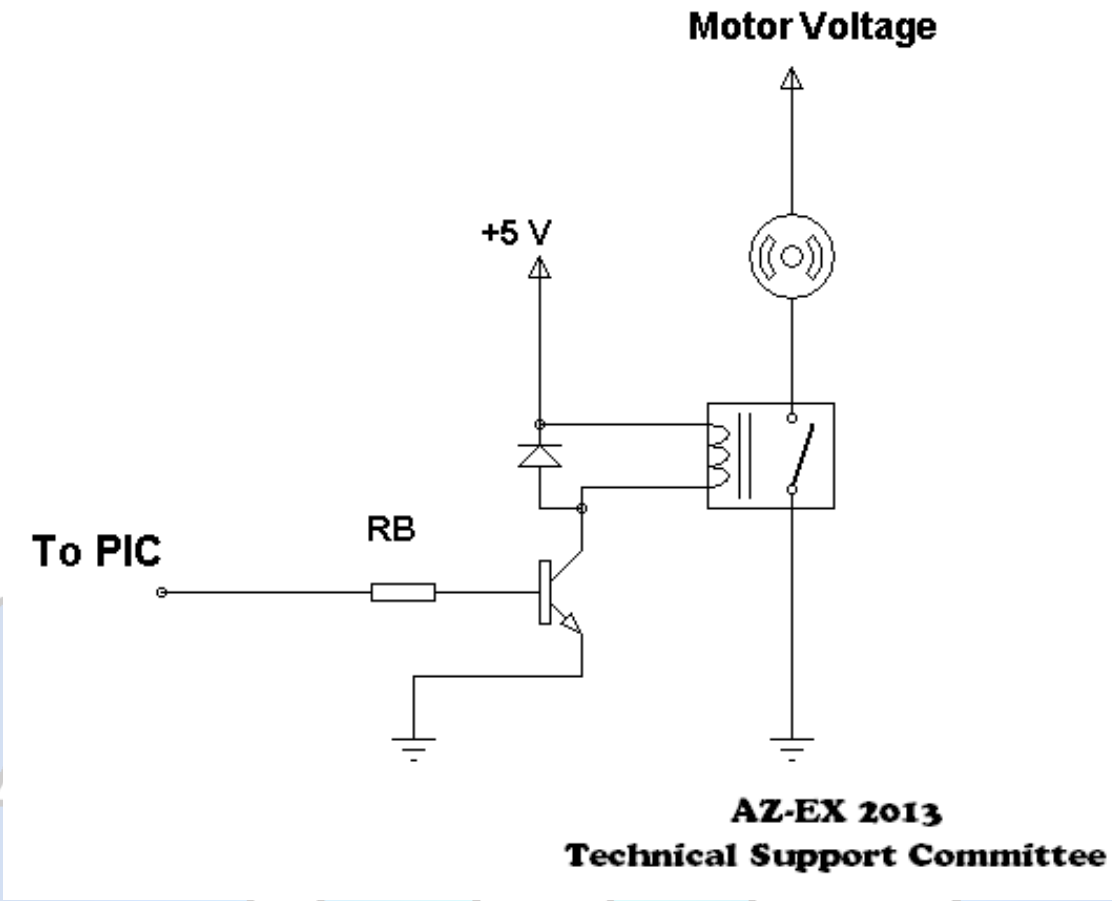
```
PORTB.B0=1;
```

ولايقاف الموتور

كود:

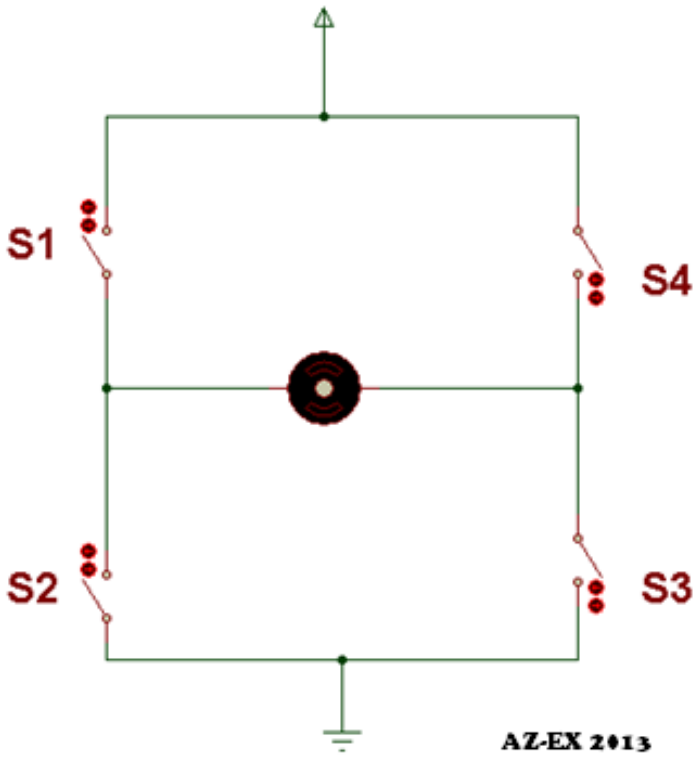
```
PORTB.B0=0;
```

ويجب استبدال الترانزستور اذا كان التيار عالى او لسبب اخر نستبدل الترانزستور بريلاى Relayكما فى الشكل ادناه

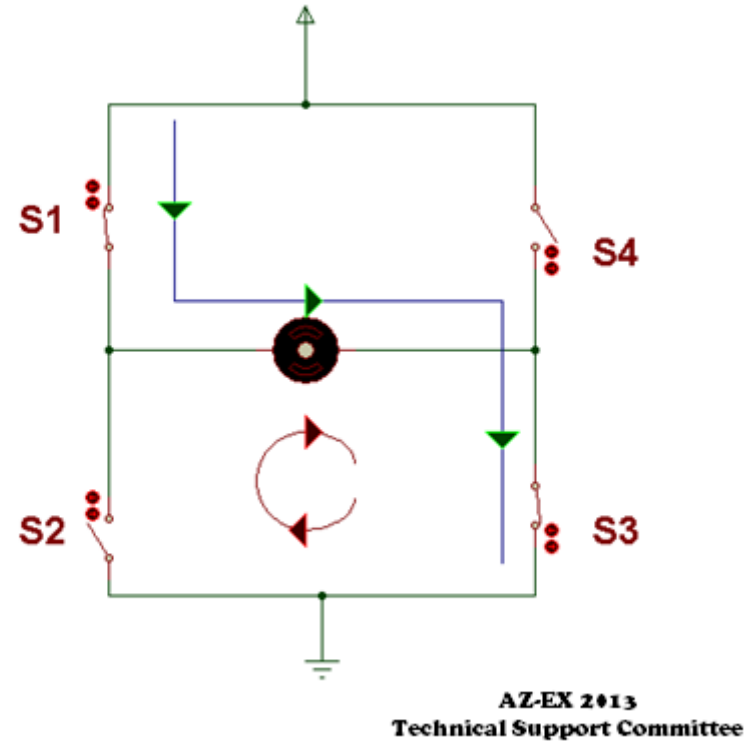


لا يسنطيع البيك توفير تيار كافى لتشغيل الريلاى لذلك قمنا بربطه مع البيك بواسطة ترانزيستور.
ملحوظة:
من الافضل عدم ربط ارضى مصدر جهد الموتور مع ارضى الدائرة المتكاملة.

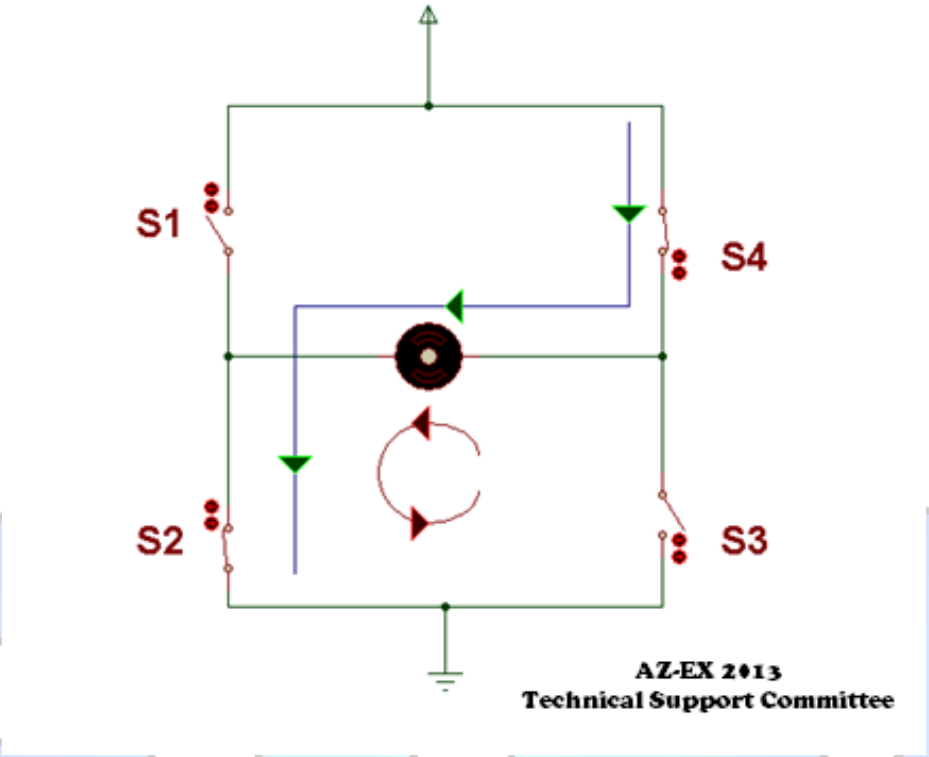
التحكم فى اتجاهين bi-direction باستخدام H-Bridge:
اذا اردنا ان نجعل الموتور يتحرك فى اتجاهينن مختلفين مرة لامام ومرة للخلف مرة فى هذا الاتجاه ومرة فى الاتجاه الاخر. بكل تاكيد الدائرة السابقة لن تسطيع هذه المقدرة على التحكم.
حتى نتمكن من ذلك نحتاج الى اربعة مفاتيح نقوم بتوصيلها كما فى الشكل التالى



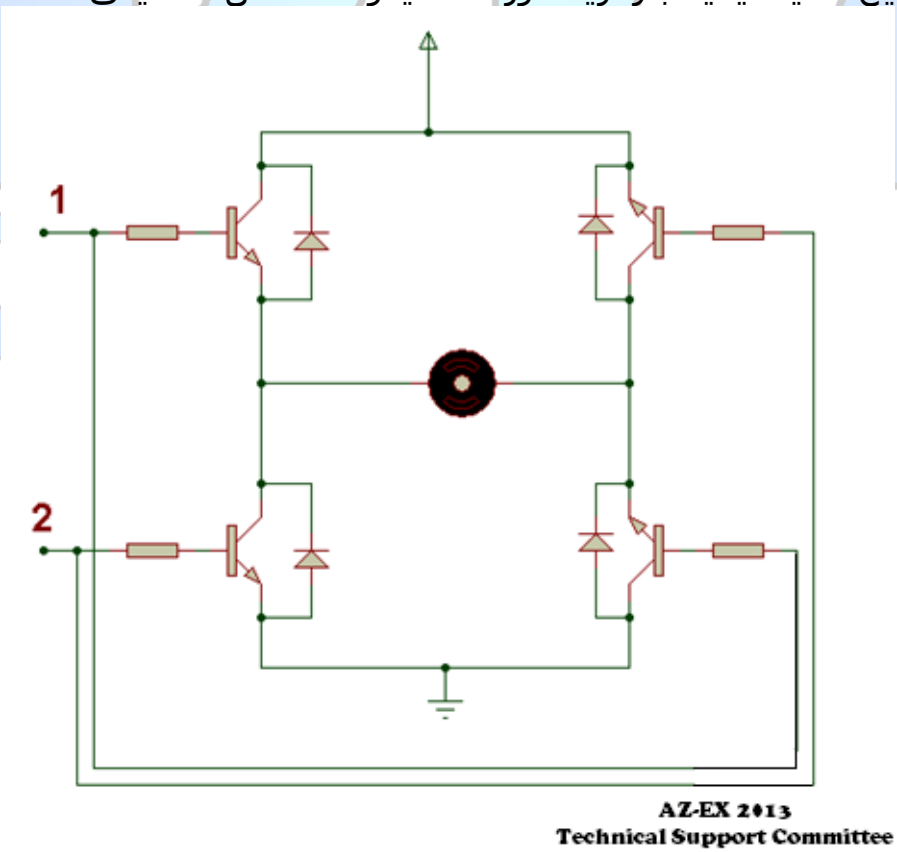
فاذا ضغطنا على المفتاحين 1 و 3 فان مسار التيار واتجاه الموتور سيكون كما فى الشكل التالى



فاذا ضغطنا على المفتاحين 2 , 4 فان اتجاه الدوران سيكون فى الاتجاه المعاكس للاتجاه الاول كما فى الشكل التالى



وباستبدال المفاتيح الميكانيكية بترانزيستورات سيكون الشكل كما يلى



فلتفرض اننا قمنا بتوصيل الطرف 1 مع PORTB.B1 و الطرف 2 مع PORTB.B2 فاذا اردنا تشغيل الموتور فى اتجاه عقارب الساعة فسنكتب

كود:

```
PORTB.B1=1;
PORTB.B2=0;
```

اما اذا اردنا تشغيل الموتور فى الاتجاه الاخر فاننا نكتب
كود:

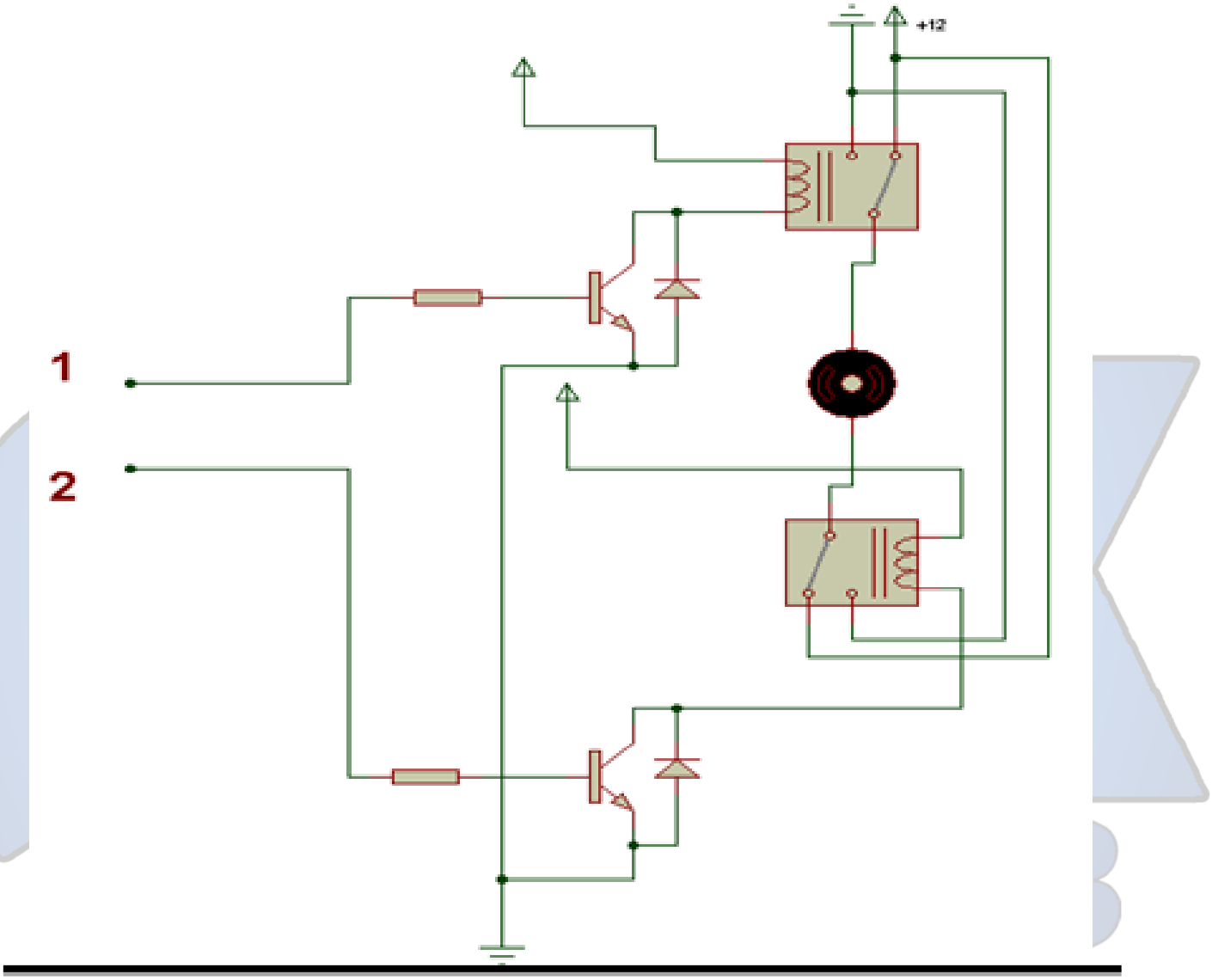
```
PORTB.B1=0;
PORTB.B2=1;
```

ولايقاف الموتور يكون الكود
كود:

```
PORTB.B1=0;
PORTB.F2=0;
```

<u>تنبيه مهم:</u>
اذا قمنا بتغذية الطرفين بموجب المصدر فان هذا سيؤدي إلى Short.

<u>طريقة أخرى لعمل H-Bridge بالريلاى:</u>
اذا اردنا استخدام الريلاى للتحكم ثنائى الاتجاه (باستخدام 2 ريلاى فقط)- فاننا نستخدم الدائرة التالية



***سادسا: التحكم في إتجاه الموتور باستخدام IC جاهزة:**

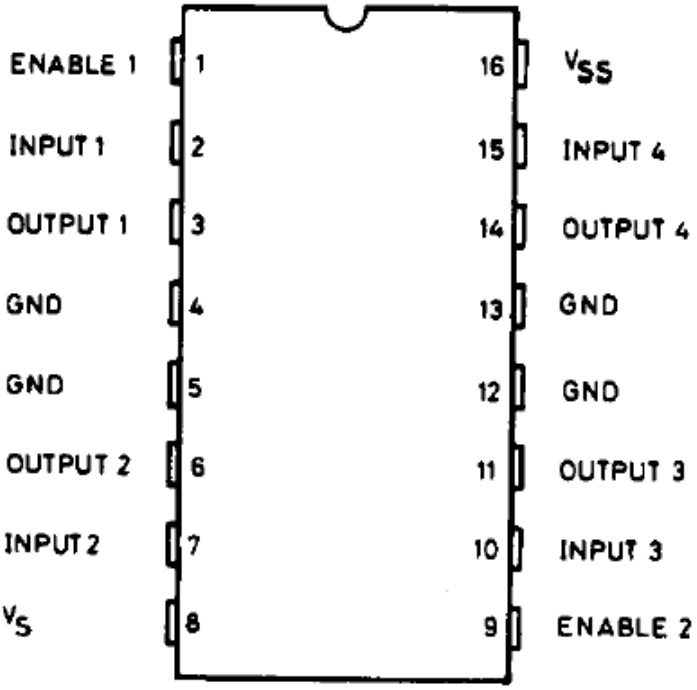
يوجد العديد من أنواع ال IC التي يمكن استخدامها في التحكم في إتجاه الموتور وعلى سبيل المثال لا الحصر سنشرح نوعين من ال IC ألا وهما L293D و L298D .

أولا : L293D :

و هذه صورتها



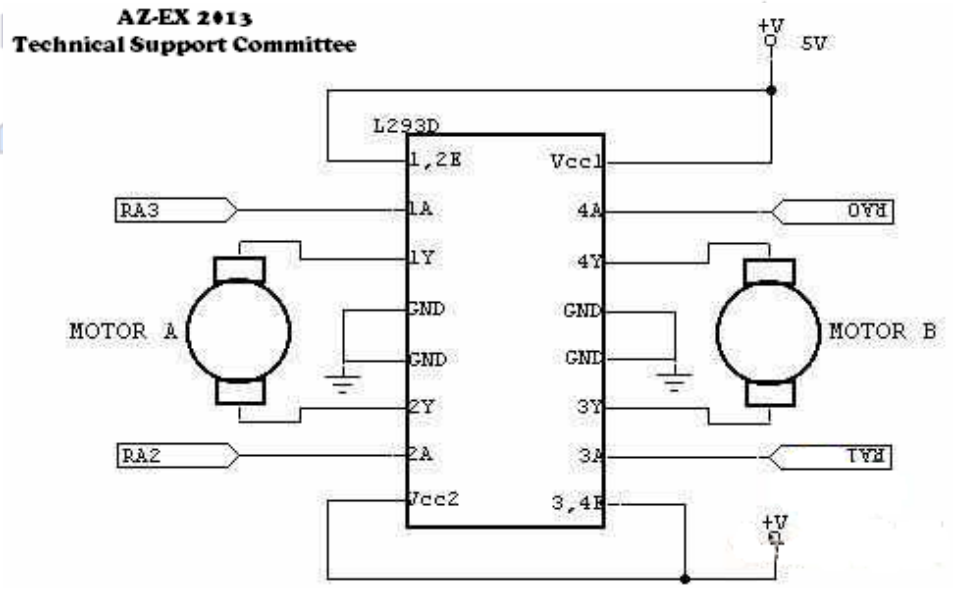
و هذا Pin Positionبتاعها



ما عملها

ال l293dاهو ic يعمل على تدوير الموتور من اليمين الى اليسار و من اليسار الى اليمين عن طريق input و ذلك بما ان المايكرو لا يعطي الى hi 5volt و low0volt

و هي توضع هكذا
انتبه هذه الصورة هي فقط للتوضيح

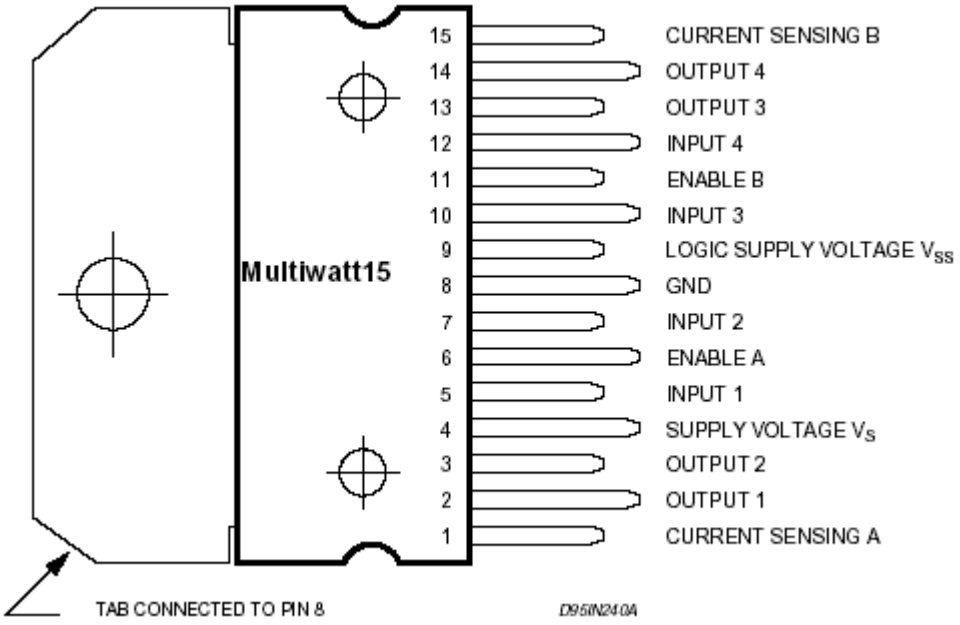


ثانيا: L298D :

و هو يعمل نفس عمل ال 293dاولكنه اقوى و لكن يجب ان نعرفه و هذه صورته



و هذا هو الpin position



ملحوظة: التطبيقات في الملفات المرفقة مع الشرح

المشروع الاول

فكرة البرنامج :

التحكم فى موتور عند طريق سوتشات عند الضغط ع السوتش الاول يدور الموتور اتجاه اليسار وعند الضغط عاى السوتش الثانى يدور اتجاه اليمين وعند الضغط على السوتش الثالث يتوقف عن الدوران .

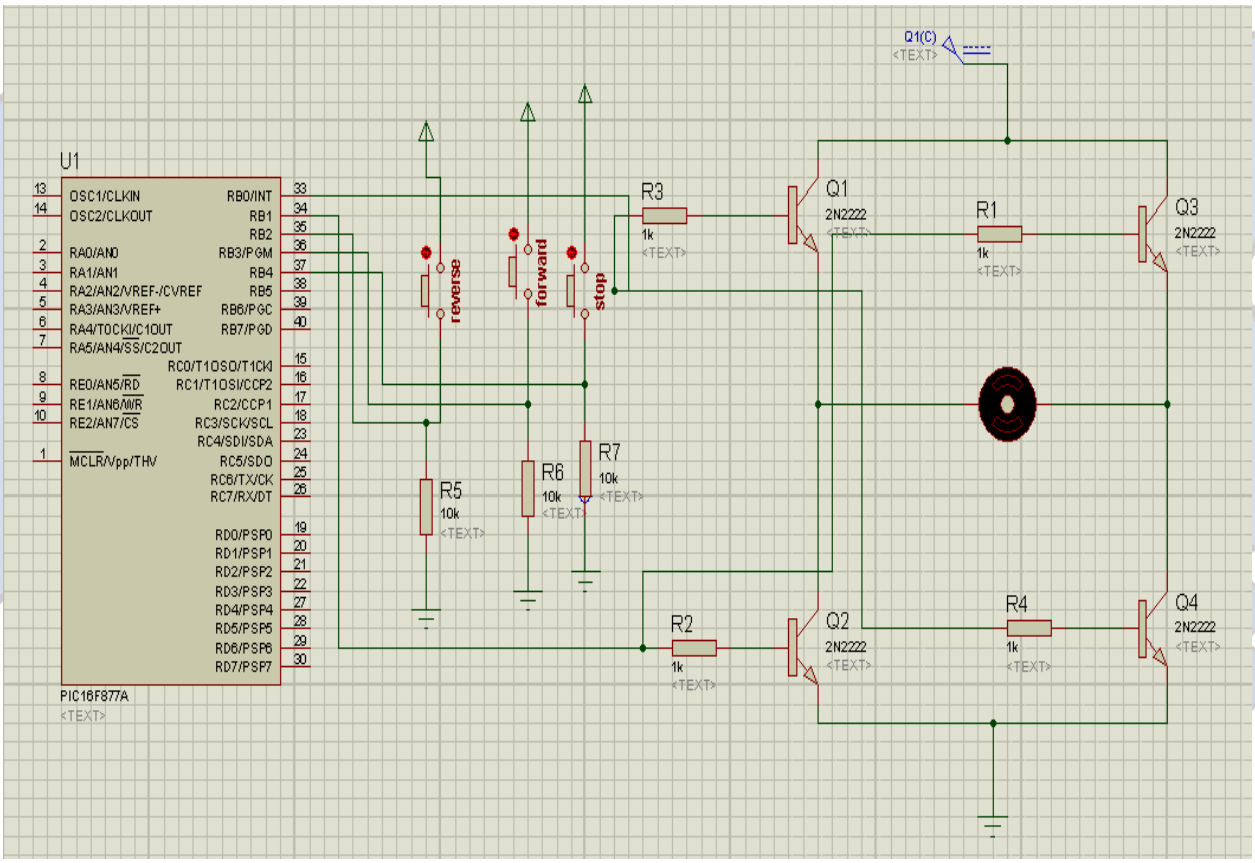
الكود :

```
void main() {
  trisb=0b00011100;                //rb0&rb1 as output / rb2&rb3 as input
  portb=0;                         // initial zero volt
  for (;;){
  if (rb2_bit==1 ){                // if button 1 pressed ,motor is reverse
       delay_ms(10);
       if (rb2_bit==1 ){
   rb0_bit=0;
   rb1_bit=1;
      while (rb2_bit==1 ){}
      delay_ms(10);
   }
   }
if (rb3_bit==1 ){  // if button 2 pressed ,motor is forward
    delay_ms(10);
if (rb3_bit==1 ){
    rb0_bit=1;
   rb1_bit=0;
   while (rb3_bit==1 ){}
      delay_ms(10);
   }
   }
  if (rb4_bit==1){  // if button 3 pressed ,motor is stop
  delay_ms(10);
  if (rb4_bit==1){
      rb0_bit=0;
   rb1_bit=0;
        while (rb4_bit==1 ){}
      delay_ms(10);
   }
       }
  }
}
```

Pin RB1&RB0 يستخدموا ك output يخرج عليهم 5v أو 0v لادارة الموتور forward أو reverse

Button	RB0	RB1
F	1	0
R	0	1
STOP	0	0

توصيل الدائرة :



المشروع الثانى

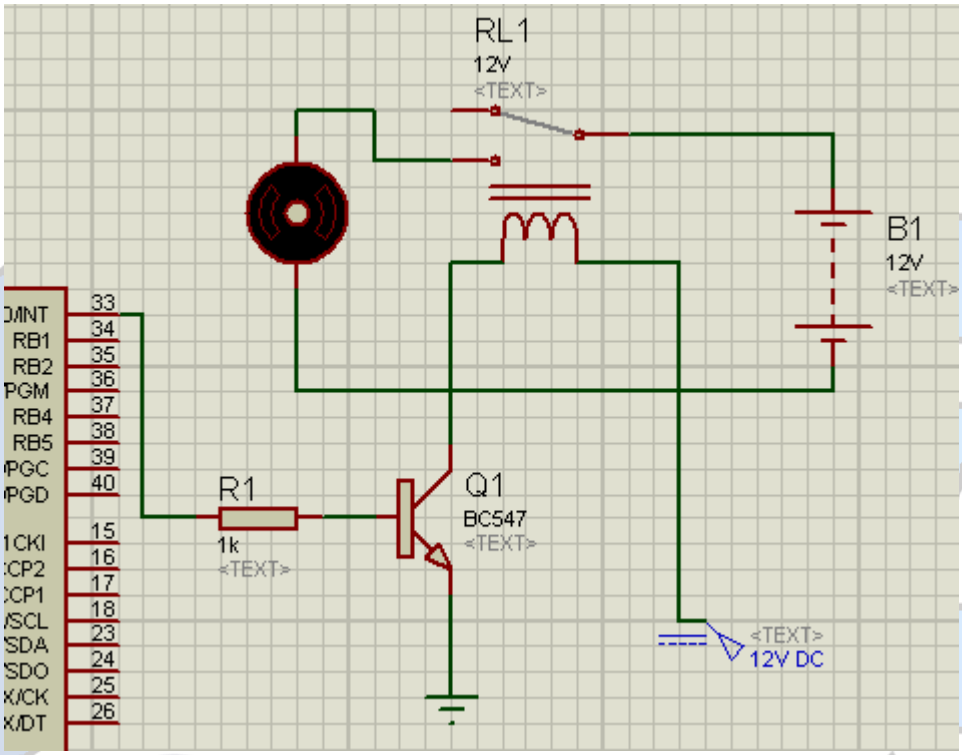
يستخدم فى هذة المشروع device مهم وهو ال relay هنستخدم نوعين :

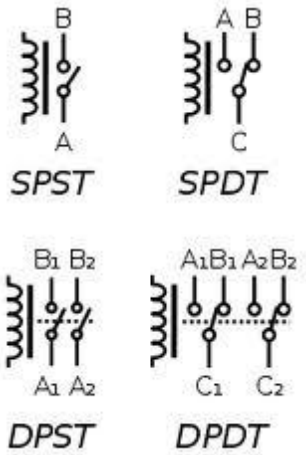
النوع الاول: SPDT relay

يعنى single pole dual throw لانه له طرف common واحد واتنين switch احدهما NC والاخر NO اى normal closed/ normal open

الفكرة لو مر تيار فى الملف يجذب التيار ال contact ويروح لل NO لو مش مر تيار يروح لل NC وبكده يعمل ON /OFF واحيانا مش بنوصل حاجه بالرجل NC وبنسبها float .

يوصل بال relay دايود فى وضع عكسى بين اطراف الملف عشان يحميه من التيارات العكسية ويوصل بب pin الميكرو ترانزستور NPN عشان اقدر اوصل بال RELAY 12 فولت دون ان يتاثر الميكرو حيث يدخل تيار صغير الى ال base من الميكرو يشغل الترانزستور فيمر تيار فى ملف ال relay يجذب ال contact الى الوضع NO فيصل 12 فولت الى الموتور فيدور





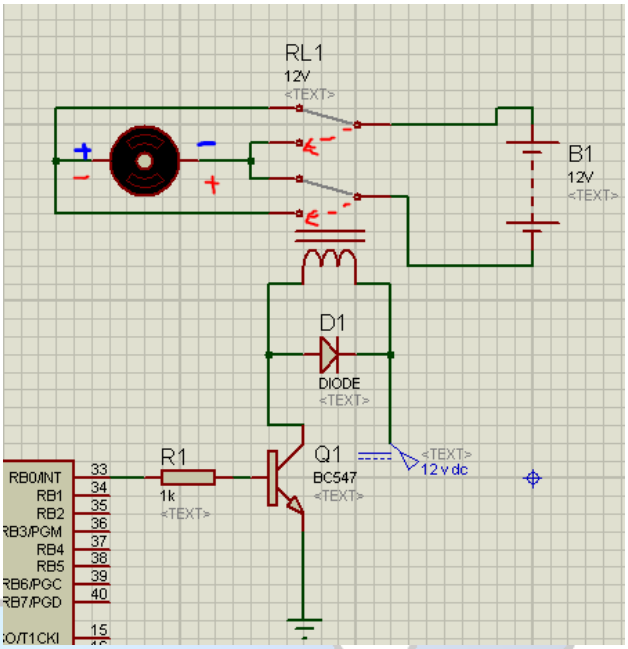
النوع التانى : DPDT

Dual pole dual throw يعنى 2 common 2 switches

هتحكم فى ال direction سواء forward or reverse عن طريق ال relay 8 pin

عندما لا يمر تيار فى الملف فان ال contact يكون على NC وتكون القطبية على اطراف الموتور كما هو موضح بالشكل والموتور يدور FORWARD يعنى بيدور forward لما مش يخرج تيار من ال pin

عندما يمر تيار فى الملف اى يخرج من الميكرو تيار ينجذب ال contact ويكون على وضع ال NO وتنعكس القطبية بين اطراف الملف كما هو موضح بالشكل ويدور الموتور فى الوضع reverse



بس انا كده اتحكمت فى ال direction بس مش بقدر اعمل on /off ؟؟؟؟

لذا ازود relay 5 pin اعمل بيه on/off عندما يخرج تيار من pin RB3 يشتغل ال relay ويكون on عندما لا يمر يكون off

RB4	RB3	Button
0	1	F
1	1	R
Don't care	0	STOP

عاوز اعمل فى الدايرة 3 Push buttons احدهما stop والثانى F والثالث R

RB0	RB1	RB2
F	R	STOP

لو عاوز ال PORTB فيه بنات OUT وبنات IN لو عاوز اجعل ال PIN OUT لازم اكتب 0 لو IN مش شرط اكتب 1 لان ال default ان ال input pins

البرنامج:

اولا عرفت port b ان اول 3pin دخل وال RB3/RB4 خرج

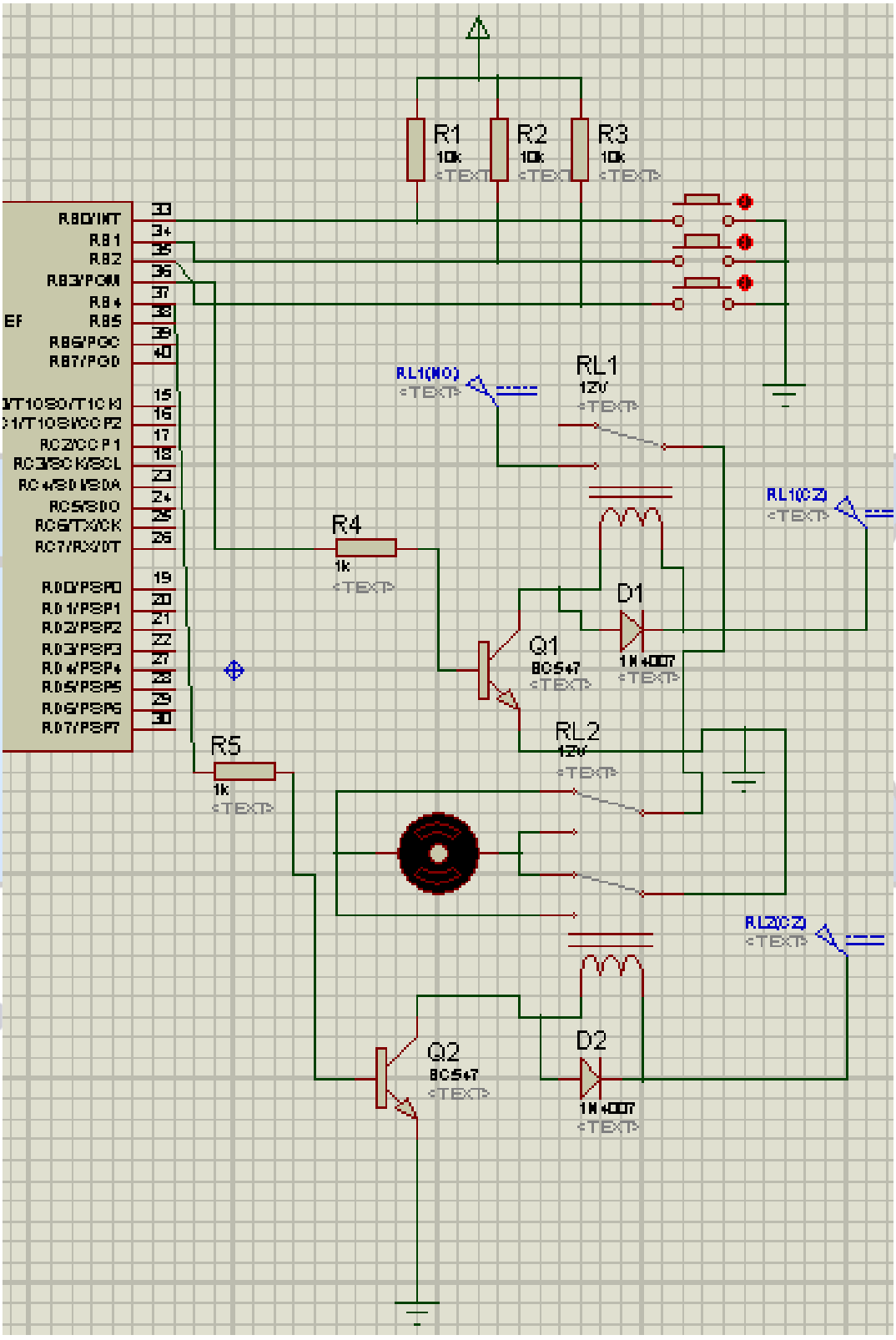
زودت شرط RB3==0 والذى لايتحقق الا عند الضغط على زر stop

عشان لو عملت forward وبعدين reverse تغير قطبية الموتور ممكن تجعلة يقف لذا اخليه مش يعمل reverse or forward الا لما اعمل stop الاول

لما اضغط على button stop اطفى ال power اجعله off بصرف النظر مر تيار فى RB4 ام لا .

```
void main (){
TRISB=0B11100111;
PORTB=0;
for(;;){
if ((RB0_bit==0)&&(RB3_bit==0)){
delay_ms(10);
if (RB0_bit==0){
RB3_bit=1;
RB4_bit=0;
while(RB0_bit==0){}
 delay_ms(10);
}
}
else if ((RB1_bit==0)&&(RB3_bit==0)){
delay_ms(10);
if (RB1_bit==0){
RB3_bit=1;
RB4_bit=1;
while(RB1_bit==0){}
 delay_ms(10);
}
}
else if (RB2_bit==0){
delay_ms(10);
if (RB2_bit==0){
RB3_bit=0;
while(RB2_bit==0){}
 delay_ms(10);
}
}
}
}
```

توصيل الدائرة :



***سابعا: التحكم في سرعة الموتور من خلال PWM:**

والآن لنقوم بشرح PWM والذي من أهم استخداماته التحكم في سرعة الموتور dc motor وكذلك التحكم في شدة الإضاءة.

أولا PWM :: اختصار لـ . Pulse Width Modulation المقصود هو إرسال موجة مربعة أو مستطيلة square wave هذه الموجة عبارة عن hi وlow .

ثانيا : يمكننا عمل موجة عبارة عن hi , low بينهم زمن انتظار معين ومن خلال هذا الزمن يمكننا التحكم في الموجة الناتجة وبالتالي التحكم في الفولت الناتج أو الطاقة الناتجة وبالتالي التحكم في سرعة الموتور .

ويمكن عمل ذلك في أي نوع من أنواع الـ pic فمثلا يمكننا كتابة الكود التالي :

```
portb.f0=1;

delay_ms(500);

portb.f0=0;

delay_ms(500);
```

الأوامر السابقة ستجعل البك يقوم بعمل موجة من الـ hi ,low
هنا سنطبق خمسة فولت من الطرف B0لمدة خمسمائة ملى ثانية ثم بعد ذلك نطبق صفر فولت لنفس المدة ثم تتكرر هذه الموجة باستمرار .
بتغيير هذا الوقت (وقت الإنتظار 500 في هذا المثال) يمكننا التحكم في الفولت الناتج فتغييرنا في هذا الرقم يقوم بتغيير ما يعرف بال duty cycle .

ما فائدة الـ duty cycle ؟؟

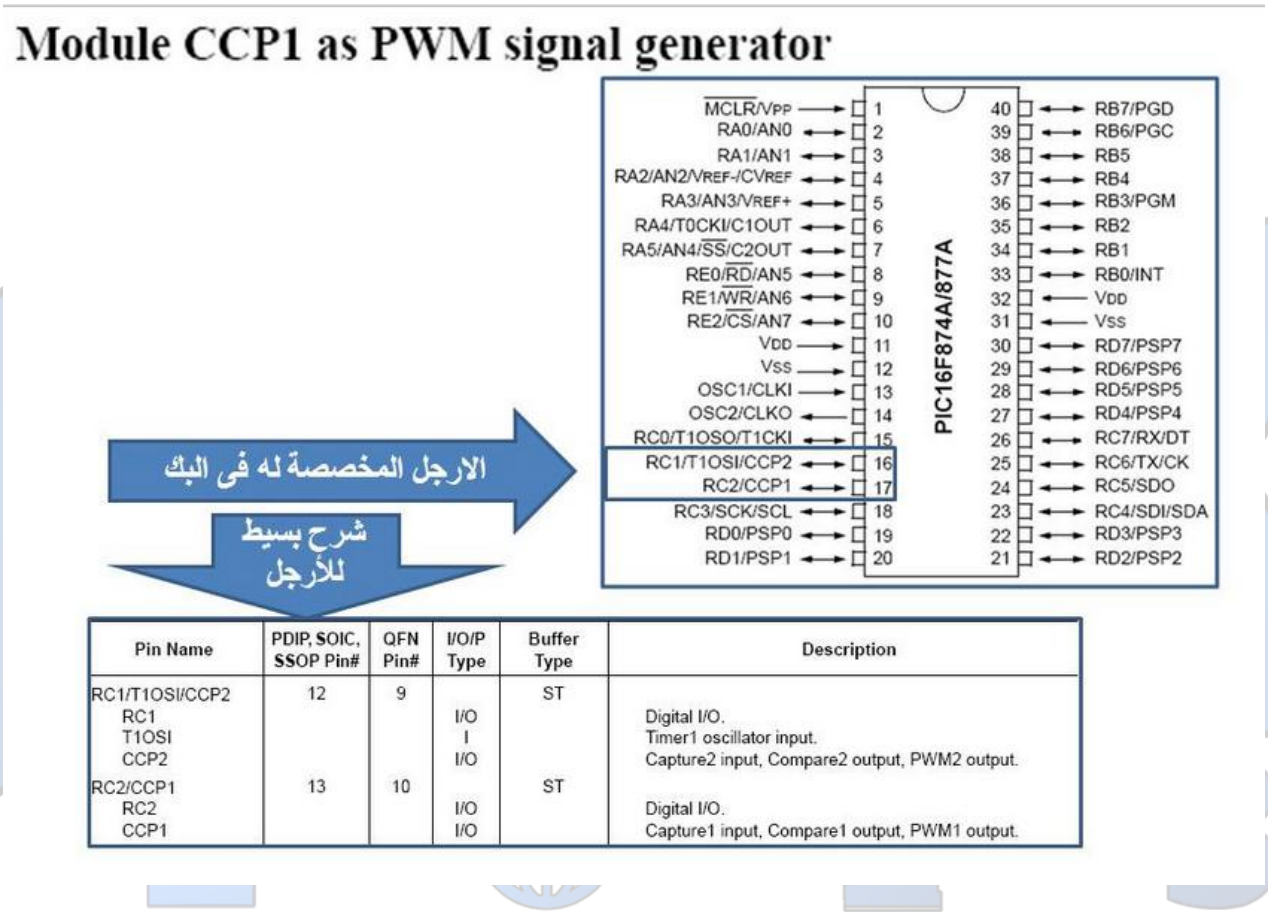
إنه هو المسؤول عن الفولت الناتج عن عملية PWM وهو عبارة عن نسبة مئوية . فمثلا لو كان لدينا جهد المصدر خمسة فولت وجعلنا الـ duty cycle بخمسين في المئة 50% فإن معدل الفولت الناتج سيكون 2.5 فولت . ولو جعلنا الـ duty cycle بخمسة وعشرين في المئة 25% فإن معدل الفولت الناتج سيكون 1.25 فولت وهكذا...

الطريقة السابقة يمكننا تنفيذها مع أي نوع من أنواع الميكروكنترولر لكننا نحتاج لمعادلات وبعض الحسابات لنعرف الفولت الناتج ولنعرف duty cycle .

ثالثا : هناك بعض الأنواع من المايكروكنترولر تدعم خاصية pwm بشكل مباشر حيث يوجد في داخل الميكروكنترولر هاردوير خاص بهذه المسألة ويوجد طرق برمجية أسهل تسهل عملية حساب duty cycle وتسهل حساب التردد . هذه الأنواع من البك pic التي تدعم هذه الخاصية

تجد أحد أطرافها مكتوب بجواره ccp1 أو . ccp2 وتمتاز هذه الأنواع من البك أنه يمكنك أن تولد موجة من أحد الأطراف في نفس الوقت الذي ينفذ فيه البك أوامر أخرى وعمليات أخرى .

وهذا طبعا لا نستطيع عمله في الأنواع التي لا تدعم pwm بشكل داخلي أهم الأنواع التي تدعم خاصية pwm بشكل مباشر والمتوفرة في الأسواق العربية
pic16f628a
pic16f877a



Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
RC1/T1OSI/CCP2	12	9		ST	
RC1			I/O		Digital I/O.
T1OSI			I		Timer1 oscillator input.
CCP2			I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	13	10		ST	
RC2			I/O		Digital I/O.
CCP1			I/O		Capture1 input, Compare1 output, PWM1 output.

شرح أوامر ال PWM

الامر الاول :

PWM1_Init(const long freq);

في بداية البرنامج نقوم بكتابة الأمر الذي من خلاله نهيئ البك لهذه العملية ومن خلال هذا الأمر يمكننا ضبط التردد بالهرتز أنظر للمثال التالي-:

Example	Initialize PWM module at 5KHz: `PWM1_Init(5000);`

الأمر السابق قام بتهيئة البك وجعل التردد الخاص بالموجة هو 5 كيلوهيرتز ويمكنك تغيير هذا الرقم (هذا التردد) على حسب الحاجة فبعض أنواع المواتير تحتاج لتردد معين مثل مواتيرالسرفو .

الأمر الثاني وهو الأهم على الإطلاق هو :

PWM1_Set_Duty(unsigned short duty_ratio);

هذا الأمر من خلاله نستطيع بكل سهولة ويسر تغيير الـ duty cycle الذي يغير الفولت الناتج

Description
Sets PWM duty ratio. Parameter duty takes values from 0 to 255, where 0 is 0%, 127 is 50%, and 255 is 100% duty ratio. Other specific values for duty ratio can be calculated as (Percent*255)/100.

Example	Set duty ratio to 75%: `PWM1_Set_Duty(192);`

والآن ماذا لو كتبنا الأمر التالي:

PWM1_Set_Duty(172);

ستكون النسبة المئوية لـ duty cycle هي خمسة وسبعون بالمئة 75 %
لماذا ؟؟
% 75 = 172/255تقريبا وبالتالي يكون الفولت الناتج عن هذا الأمر لو كان الجهد المصدري لنا خمسة فولت ... 75. ضرب 5 = 3.75 فولت.

الامر الثالث :

عليك أن تنتبه أن الموجة المطلوبة لن تخرج إلا عن طريق الأمر الذي يبدأ هذه العملية وهذا الأمر هو:

PWM1_Start(void);

Example	`PWM1_Start();`

بعد أن نقوم ببدء هذه العملية ال pwm يمكننا أن نقوم بتغيير ال duty cycle لتغير الفولت والذى بدوره يقوم بتغيير سرعة الموتور .

الامر الرابع :

في النهاية : إذا أردنا أن نوقف هذه الموجة ولا نخرج أي جهد من الطرف ccp1 علينا أن نكتب الأمر التالي :

PWM1_Stop();

Example	PWM1_Stop();

معلومة إضافية مهمة : بعض أنواع المتحكمات (البك) تحتوي على اثنين PWM

وبالتالي يمكننا التحكم من خلالهما في موتورين كل موتور بسرعة معينة كما نريد ومن

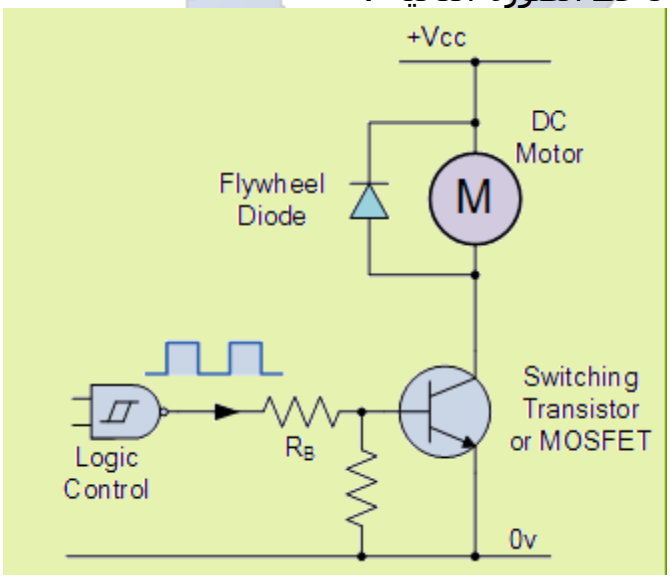
الأمثلة على هذه المتحكمات . pic16f877a فهو يحتوي على طرفين
ccp1
و
ccp2
يمكنك أن توصل بكل طرف منهم موتور يعمل بسرعة مستقلة عن الآخر.

ولكن كيف سأقوم بعمل ذلك في البرمجة :

الأمر بسيط جدا في لغة مايكروسي إن الأوامر السابقة التي كتبنها ستجعل البك يستخدم الطرف CCP1 أما إذا أردنا استخدام ccp2 فقط قم باستخدام نفس الأوامر ولكن مع إضافة الرقم اثنين بدلا من واحد.

الملاحظة الثانية يجب أن نقوم بتوصيل دايود مع الموتور للحماية من التيار العكسي الناتج عن خاصية الحث الذاتي وهذا الدايود نستخدمه في أي تطبيق به ملف مثل الريلاي والموتور .

لاحظ الصورة التالية :



أيضا من الخطأ التحكم في سرعة الموتور عن طريق تشغيل واطفاء الريلاي (أقصد الريلاي العادي وليس الـ reed relay وذلك لأننا نقوم بتوليد نبضات سريعة تشغيل واطفاء تشغيل واطفاء وللأسف سرعة استجابة الريلاي العادي بطئية جدا فربما لا يستطيع أن يعمل وينطفئ إلا مرات معدودة في الثانية الواحدة فبالتالي لن يكون مناسب في هذا التطبيق لذا نستخدم الترانزستور سواءا كان BJT أو MOSFET لأن سرعة استجابته عالية جدا .

المشروع الثالث

فكرة المشروع :

التحكم فى سرعة موتور عن طريق سوتشات

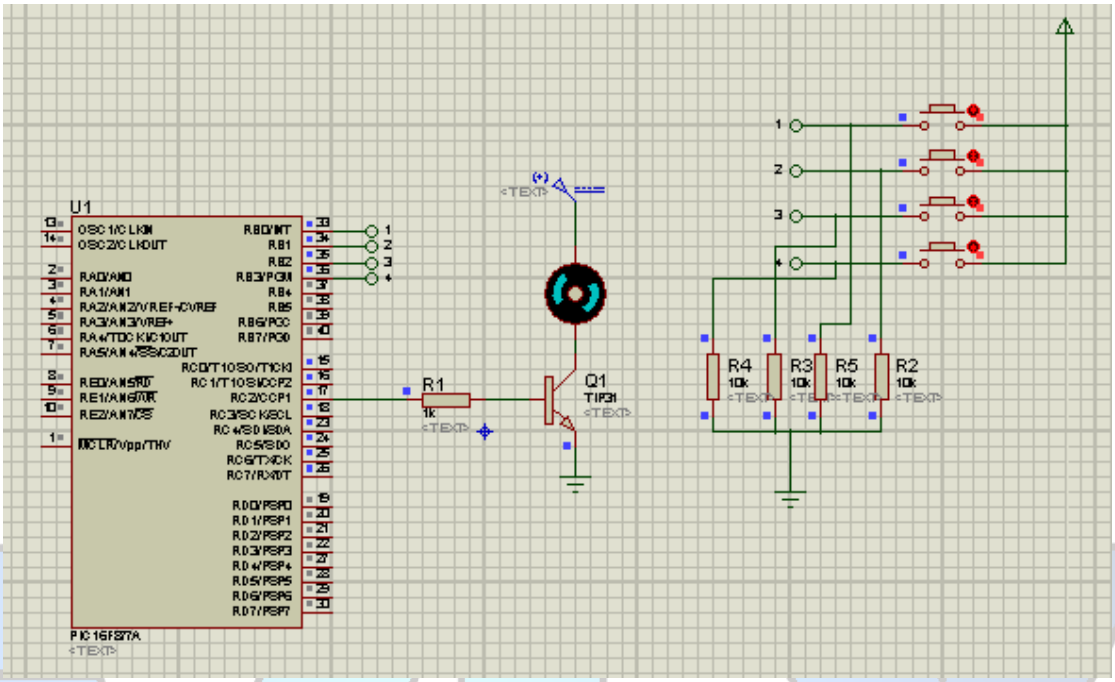
الكود :

فى الكود بقوله لو ضغطت على المفتاح الاول خلى الموتور يدور بسرعة معينة لو ضغطت على السوتش التانى اقلل السرعة شوية لو ضغطت على السوتش الثالث قلل السرعه أكتر لو ضغطت على السوتش الرابع وقف الموتور عن طريق وقف ال pulses الخارجة من الميكرو

```
void main() {
trisb=255; //portb is input
trisc=0; // portc is output
portc=0;
pwm1_init(400);
while(1){
if(rb0_bit==1){
pwm1_start();
pwm1_set_duty(255);
 }
 if(rb1_bit==1){
 pwm1_start();
 pwm1_set_duty(200);
 }
 if(rb2_bit==1){
 pwm1_start();
 pwm1_set_duty(100);
 }
 if(rb3_bit==1){
 pwm1_stop();
   }
 }

}
```

توصيل الدائرة :



المشروع الرابع

فكرة المشروع :

التحكم فى سرعة متورين عن طريق السوتشات

الكود :

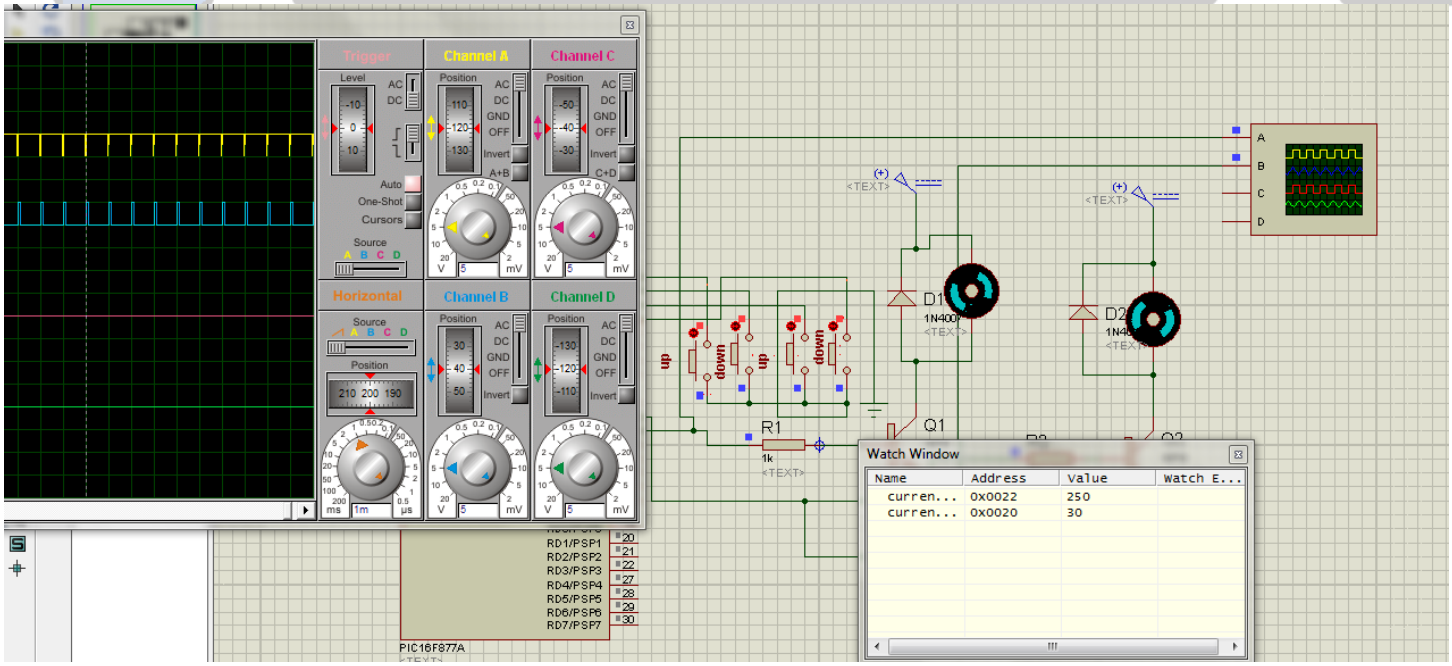
فى الكود استخدم الامر not_rbpu_bit=0 من المعروف ان portb بداخلة pull up resistor فبدل من توصيل مقاومات باستخدام هذا الامر لتنشيط هذة المقاومات , ثانيا قمت بتعريف متغيرين current_duty خاص بالموتور الاول و متغير current_duty1 خاص بالموتور الثانى عند الضغط على المفتاح الاول تزداد قيمة المتغير current_duty وبالتالى تزداد ال duty cycle ثم تزداد سرعة الموتور الاول وعند الضغط على المفتاح الثانى تقل قيمة المتغير current_duty وبالتالى تقل ال duty cycle ثم تقل سرعة الموتور الاول

وهكذا بالنسبة للمتغير current_duty1 لكن لاحظ ان ال duty cycle ratio يتراوح من 0_255
يعنى لازم اضيف شرط زيادة لوضغطت على المفتاح الاول وكان ال current_duty<250 يزود
السرعة لو ال لو ال current_duty>250 مش يعمل حاجة تفضل سرعة الموتور زى ما هيا
حتى لو ضغطت على المفتاح up والذى يزيد من سرعة الموتور وهكذا لو ضغطت على
المفتاح الثانى وكان ال current_duty>30 فانة يقلل سرعة الموتور أما لو ال
current_duty<30 مش يعمل حاجة تفضل سرعة الموتور زى ما هيا وهكذا بالنسبة للموتور
الثانى .

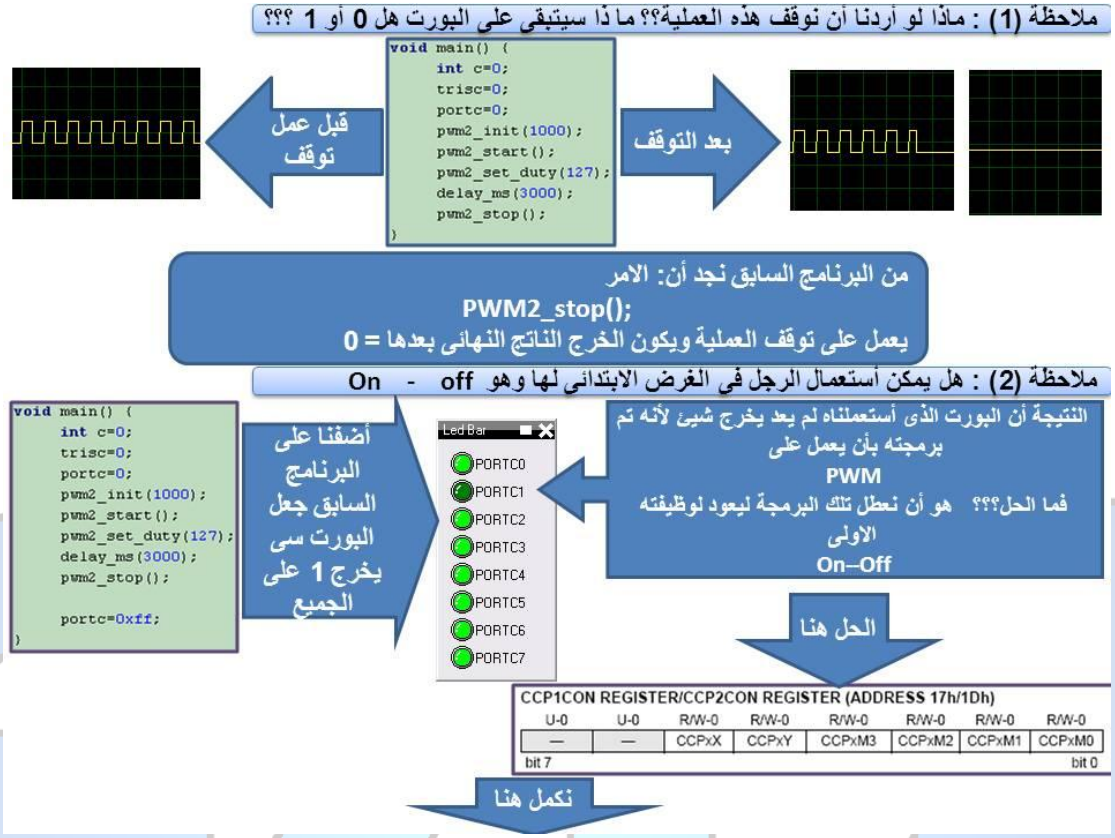
```
int current_duty=0,current_duty1=0;
void main() {
    TRISB = 255;                          // configure PORTB pins as input
    not_rbpu_bit=0;
  PWM1_Init(1000);                       // Initialize PWM1 module at 5KHz
  PWM2_Init(1000);                       // Initialize PWM2 module at 5KHz
  current_duty  = 30;                    // initial value for current_duty
  current_duty1 = 30;                    // initial value for current_duty1
  PWM1_Start();                          // start PWM1
  PWM2_Start();                          // start PWM2
  PWM1_Set_Duty(current_duty);           // Set current duty for PWM1
  PWM2_Set_Duty(current_duty1);          // Set current duty for PWM2
  while (1) {                            // endless loop
    if (rb0_bit==0 && current_duty<250 ) {                    // button on RB0 pressed
      delay_ms(10);
       if (rb0_bit==0) {
      current_duty=current_duty+5;                 // increment current_duty
      PWM1_Set_Duty(current_duty);
        while (rb0_bit==0) {}
      delay_ms(10);
       }
```

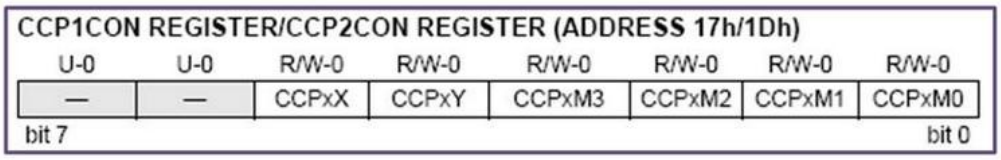
```
    }
   if (rb1_bit==0 && current_duty>30) {                    // button on RB1 pressed
     delay_ms(10);
  if (rb1_bit==0) {
     current_duty=current_duty-5;                        // decrement current_duty
     PWM1_Set_Duty(current_duty);
     while (rb1_bit==0) {}
      delay_ms(10);
      }
    }
   if (rb2_bit==0&& current_duty1<250) {                   // button on Rb2 pressed
    delay_ms(10);
  if (rb2_bit==0) {
     current_duty1=current_duty1+5;                       // increment current_duty1
     PWM2_Set_Duty(current_duty1);
     while (rb2_bit==0) {}
      delay_ms(10);
      }
    }
   if (rb3_bit==0 && current_duty1>30) {                   // button on Rb3 pressed
  delay_ms(10);
  if (rb3_bit==0) {
     current_duty1=current_duty1-5;                       // decrement current_duty1
     PWM2_Set_Duty(current_duty1);
       while (rb3_bit==0) {}
      delay_ms(10);
      }
    }
   }
}
```

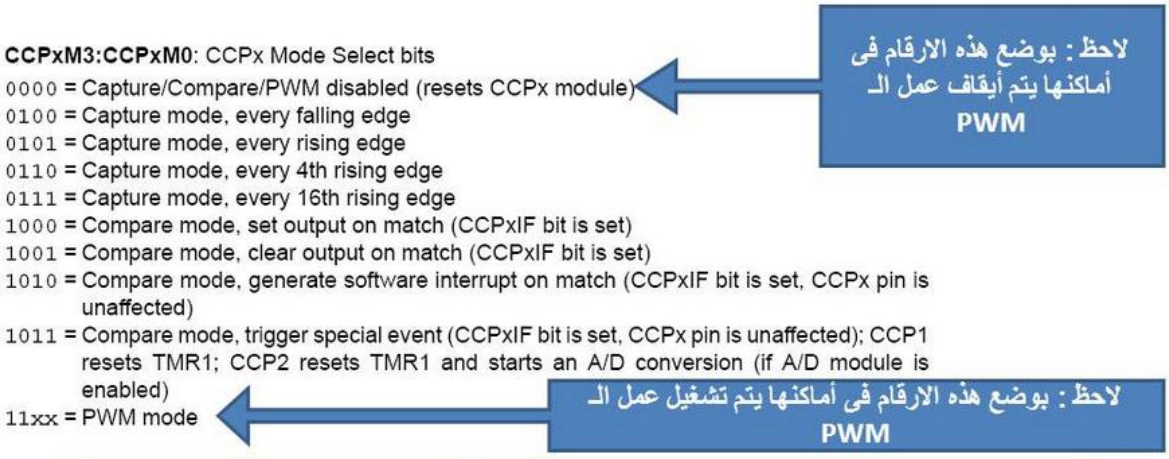
توصيل الدائرة :



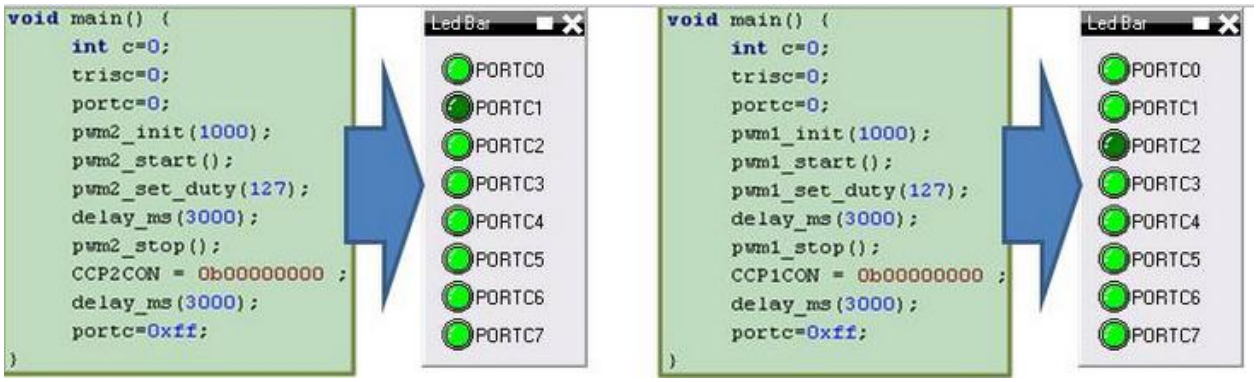
لاحظ :



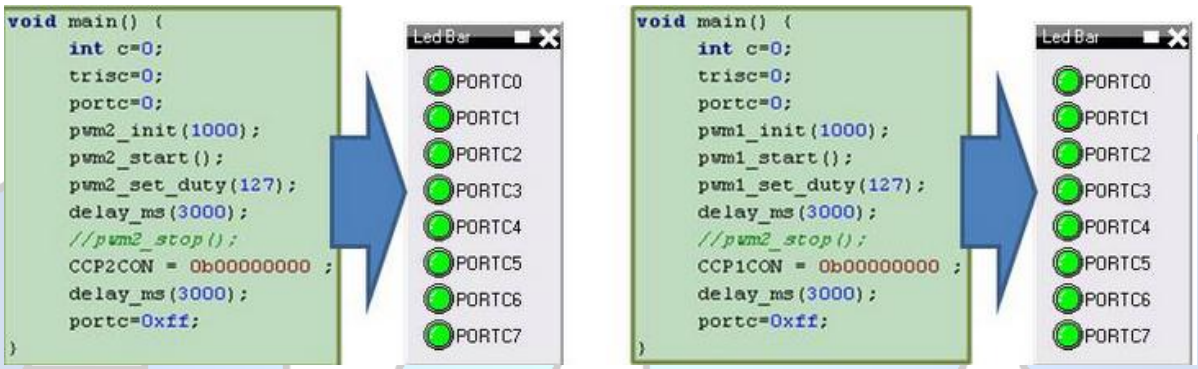




لذا فانه لايقاف هذة ال PWM فانه لا نستخدم الامر pwm1_stop (); لانه يجعل خرج ال pin=0 فلو وصلت عليها ليد لا يمكن اضاءتها ألا بأكواد أخرى فبدل أستخدام الكود التالى لان الامر pwm_stop يجعل خرج ال pin =0



نستخدم الكود التالى



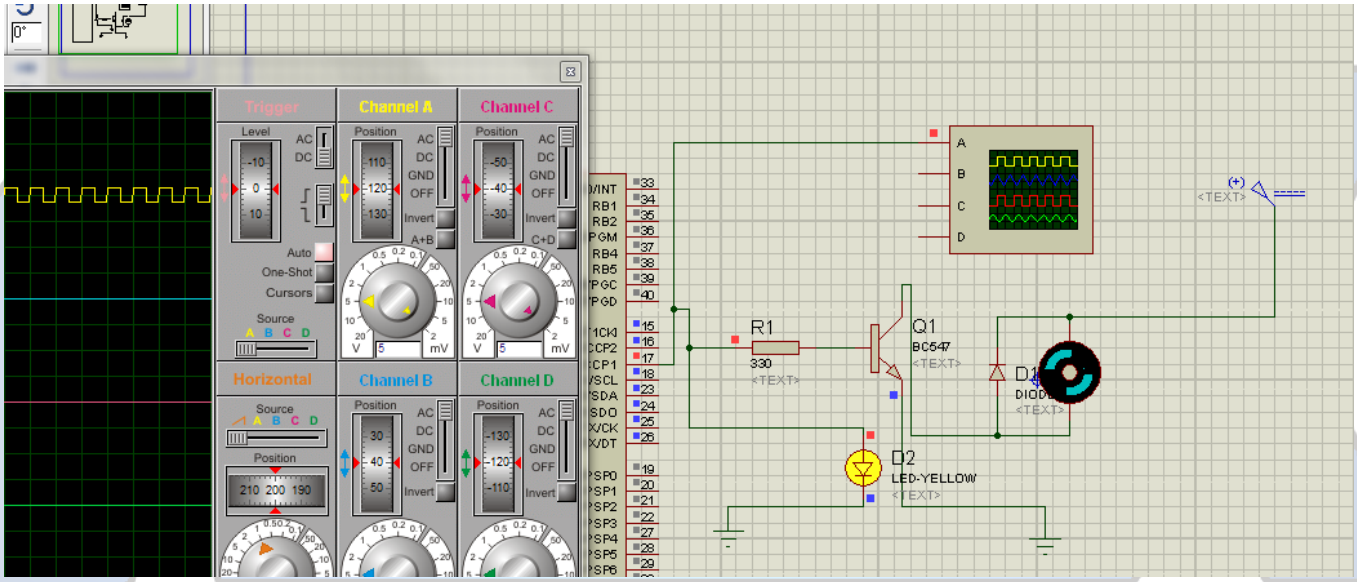
مثال

فى الكود التالى

```
int i;
sbit led1 at portc.b2;
void increse(){
for(i=0;i<=255;i++){
pwm1_init(1000);
PWM1_start();
pwm1_set_duty(i);
delay_ms(50);
//pwm1_stop();
ccp1con=0b00000000;
}
}
void led(){
led1=1;
}
void main() {
trisc=0;portc=0;
while(1){
increse();
led();
 delay_ms(5000);
}
}
```

نلاحظ عند عمل سميوليشن للبرنامج أن سرعة الموتور تزداد وذلك لانه فى بداية البرنامج أى بداية ال main يذهب للدالة increase والتى تتحكم فى ال duty cycle عن طريق متغير i تزداد قيمته مع كل عدة وبالتالى تزداد سرعه الموتور وتقوم الليد بعمل فلاشر بسبب الموجة المربعة الخارجة من pin ال pwm لحد ما يوصل المتغير I الى القيمة 255 وتنتهى ال for loop وقتها هيروح للدالة led ويشغل الليد لمدة 5 ثوانى وهيكون بردة وقتها الموتور بيلف بأقصى سرعة لان اخر قيمة لل i=255 .

توصيل الدائرة :



Chapter (8)

ADC



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

بسم الله الرحمن الرحيم.....
إن شاء الله تعالى سنتعلم في هذا الدرس كيفية تحويل القيم ال Analogإلى Digital عن طريق الميكرو كنترولر، واستخدامها في التطبيقات المختلفة كقياس درجة الحرارة و غيرها من التطبيقات.

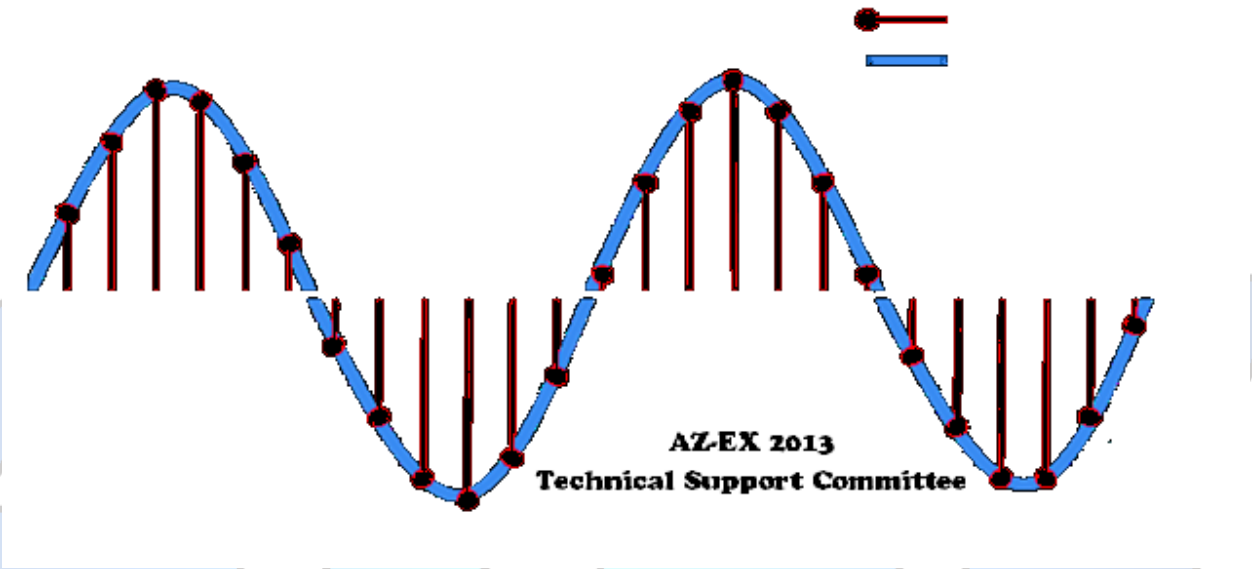
سنتناول في هذا الدرس إن شاء الله:
1-لماذا نستخدم ADC؟
2-مبدأ عمل ADC.
3-الأرجل مخصصة لل ADC في Pic16F877A.
4-الأوامر المسخدمة في MicroC لعمل ADC.
5-بعض التطبيقات على الADC.

***أولا: لماذا نحول القيم التناظرية (Analog) إلى رقمية (Digital) ؟**
بعد تطور الانظمة العددية المختلفة وما رافقها من تطور في الاجهزة الالكترونية فقد ظهرت انظمة الحاسوب بشكل عام ، وقد كانت بداية تستخدم لأداء عمليات حسابية بسيطة جدا وتستغل حجما كبيرا جدا .. ومع التطور المتواصل أدى ذلك الى زيادة استخدام الحاسوب في مجالات الحياة المختلفة ،مثل استخدامه في أنظمة جمع المعلومات ، فمثلا يمكن استخدام الحاسوب ليراقب درجة حرارة فرن حراري بحيث اذا قلت عن درجة معينة يقوم بزيادة الحرارة في الفرن ، طبعا إن عملية القياس تتم بواسطة حساسات (سنسورات) وظيفتها تحويل درجة الحرارة الى جهد كهربائي متغير حسب درجة الحرارة داخل الفرن ،لذلك يجب مراقبة درجة حرارة الفرن باستمرار ،حيث يتم اخذ قراءات درجة الحرارة كل فترة زمنية معينة ومقارنتها مع الدرجة المعيارية ومن ثم اتخاذ الاجراء المناسب ، طبعا الحاسوب لا يستطيع الا التعامل مع الاشارات الرقمية ، بينما يكون جهد خرج السنسورات انالوج او ديجيتال .. في حال كان جهد خرج السنسور ديجيتال (رقمي) فلا مشكلة في ذلك حيث يتم تغذية الاشارة من السنسور الى الحاسوب مباشرة ، اما في حال كانت اشارة السنسور انالوج فاننا نحتاج الى قطعة تقوم بتحويل الاشارة الانالوج الى ديجيتال ، هذه القطعة (او الدارة تدعى بالمحول من انالوج الى ديجيتال ADC) طبعا في السابق كنت تحتاج الى قطعة منفصلة تربط مع البيك (او الحاسوب) ليستطيع البيك التعامل مع القراءات المختلفة . اما الان فقد تم دمج هذه الخاصية او المحول في البيك واصبحت من مميزاته مما يوفر الكلفة الكلية وخطوط الربط بين الدارات وهذا سيكون موضوعنا.

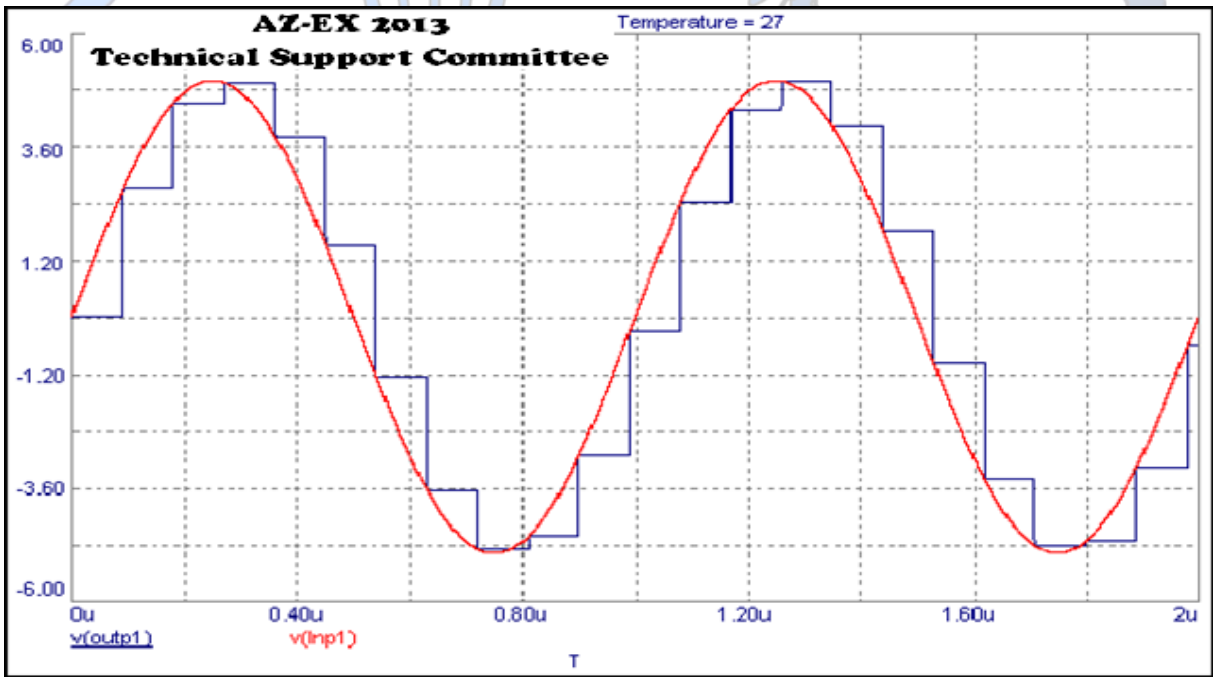
***ثانيا: مبدأ عمل ADC.**
يتلخص عمل ADC من أنالوج إلى ديجيتال بالتالي:
1-يقوم في البداية بأخذ عينات من الإشارة الأصلية ويثبتها خلال فترات زمنية ثابتة..
2-يتم تقسيم المسافة على كامل مجال الإشارة المطلوبة إلى مستويات ثابتة..
3-يتم تحديد المستوى الذي تنتمي إليه الإشارة عند العينة المطلوبة.
4-يتم إخراج قيمة المستوى المطلوب على مخارج الADC.

كما نلاحظ فان عمل الADC يتلخص في مجموعة خطوات بسيطة (لا تقلق إذا لم تفهم ما تعنيه كل خطوة لأننا سنتحدث عن كل خطوة بالتفصيل).. الآن سنبدأ شرح كل خطوة بالتفصيل:

1- أخذ العينات من الاشارة الاصلية وتثبيتها(Sampling) : في بداية عملية التحويل يتم اخذ عينات(Samples) تعبر في شكلها عن الاشارة الاصلية ... كما في الشكل التالي :

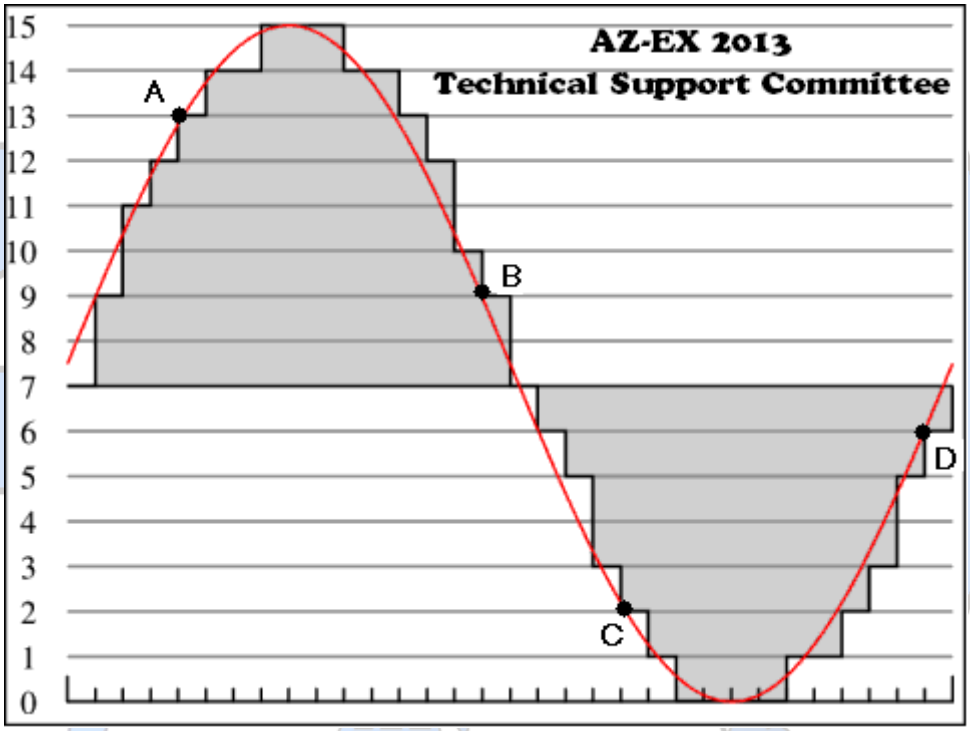


الان يتم اخذ العينة وتثبيت القيمة لهذه العينة خلال فترة عملية التحويل كما في الشكل التالي:



كلما زاد عدد العينات المأخوذة نلاحظ اقتراب شكلها اكثر من الاشارة الاصلية وهذا يسمى (Sampling Frequency)..طبعا اذا قل عدد العينات عن عدد معين فان الاشارة سيحدث لها تشويه وستختلف عن الاشارة الاصلية لذلك يجب ان يكون عدد العينات كافي ليشابه الاشارة الاصلية.

2-تقسيم المسافة على كامل مجال الاشارة الى مستويات (Quantization) : الان بعد ان تم تثبيت قيمة العينة خلال فترة التحويل فانه يتم تقسيم المجال العمودي الى مستويات ثابتة كما في الشكل التالي:



3- نلاحظ ان الشكل السابق مقسم الى 16 مستوى عمودي (من 0 – 15)حيث يتم تحديد العينة ورقم المستوى المقابل لها ..فكما نرى مثلا ان العينة (A) لها قيمة مستوى مساوية لـ 13 ، والعينة (B) لها قيمة مساوية لـ 9 ، والعينة (C) لها قيمة مساوية لـ 2 ، والعينة (D) لها قيمة مساوية لـ 6.

4-الان يتم اخراج قيمة المستوى المكافىء على اطراف خرج المبدل من تماثلي الى ديجيتال.

الان لو سألتك سؤالا : ما نوع ADC السابق ، هل هو 16 بت كما رأيت من الشكل السابق ؟؟؟

طبعا لا فليست هذه طريقة الحساب بالنسبة للADC ... بما اننا نتعامل مع النظام الرقمي(الديجيتال) هنا في خرج الADC فان عملية التعبير يجب ان تكون تابعه لنظام الديجيتال .وبالتالي نقوم بمعرفة عدد مستويات خرج الADC وهي هنا في حالتنا 16 مستوى وبالتالي لنعرف نوع الADC نستخدم العلاقة التالية

$$2^{n} = 16$$

AZ-EX

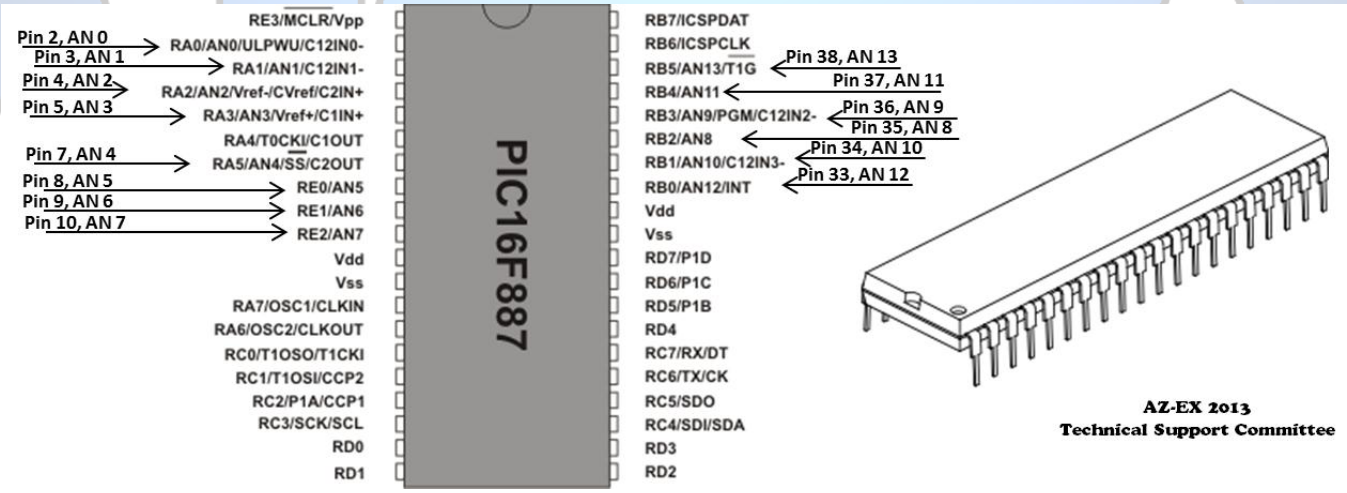
حيث يعبر الرقم 2 عن النظام الرقمي ... اما n عدد البتات للADC ... اما الرقم 16فهو عدد المستويات وفي حال كان عدد المستويات غير ذلك نقوم باستبدال الرقم 16 في العلاقة السابقة بالعدد الجديد ونحسب اعتمادا عليه..

الان نستطيع بسهولة حساب حسب عدد البتات للADC من العلاقة السابقة وهي تساوي n=4بت ، يمكن حساب عدد بتات خرج أي ADC من العلاقة السابقة بمعرفة عدد مستوياته ، او بالعكس يمكن معرفة عدد مستويات الADC من خلال عدد البتات للخرج (علاقة تبادلية سهلة).

ملاحظة : بزيادة عدد مستويات الخرج للADC (أي زيادة عدد بتات الخرج) فانه يصبح أكثر دقة لأن الاشارة تقترب اكثر من الشكل الاصلي.

***ثالثا: الأرجل مخصصة لل ADC في Pic16F877A.**

هذه الصورة توضح الأرجل المسؤولة عن ال ADC :



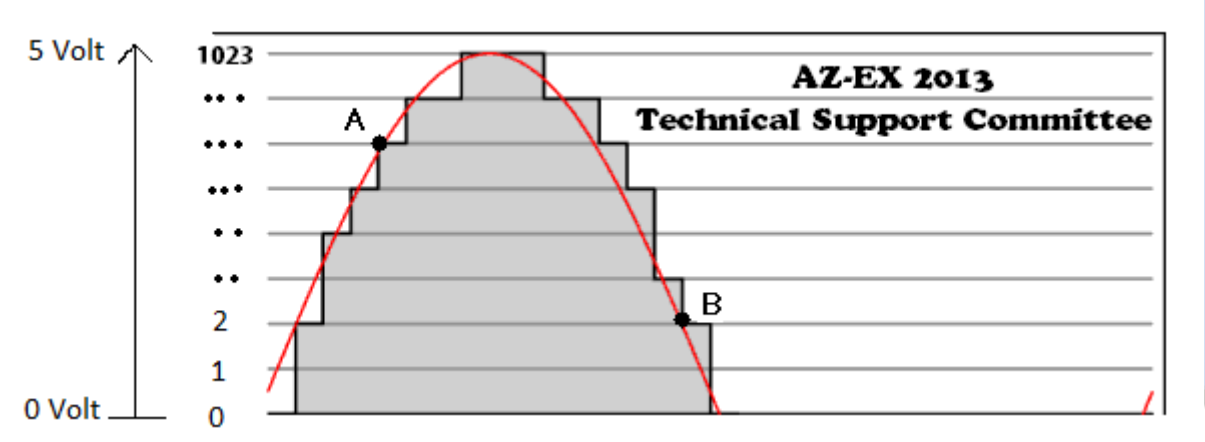
***رابعا: الأوامر المسخدمة في MicroC لعمل ADC :**

سنتعرف الآن على برنامج بسيط لقراءة فولت من AN 0 وتحويله إلى Digital وتسجيله في متغير.

```
int num;
float Volt;
num=adc_read(0);
Volt=num*(5/1023);
```

الأمر adc_read(); يقوم بقراءة الفولت وتقسيمه إلى samples ثم يقوم بعملية ال Quantization ويحدد قيمة الفولت بعدد معين يخزن في المتغير "num"

ولتحويل هذا العدد إلى القيمة الحقيقية للفولت نقوم بعملية حسابية وهي ضرب هذا الرقم في 5 وقسمته على 1023 ولتوضيح هذه النقطة ننظر للشكل الآتي:



من الواضح أن القيمة العظمى التي تنتج من ال Quantization هي 1023 ، والقيمة العظمى للفولت الداخل على ال AN 0 هو 5 فللتحويل من النظام الأول للثاني نضرب * 5 ونقسم \ 1023

وبذلك تم حفظ القيمة الحقيقية للفولت في المتغير "volt"

***خامسا: بعض التطبيقات على الADC.**

التطبيقات في الملفات المرفقة مع الشرح

Chapter (9)

Keypad



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013



بسم الله الرحمن الرحيم.....

إن شاء الله تعالى سنتعلم في هذا الدرس كيفية توصيل ال keypad مع الميكرو كنترولر ، وكيفية إدخال الأرقام إلى الميكرو كنترولر عن طريق ال keypad وإمكانية عرضها على LCD.

سنتناول في هذا الدرس إن شاء الله:

1- ممَّ تتكون ال Keypad.

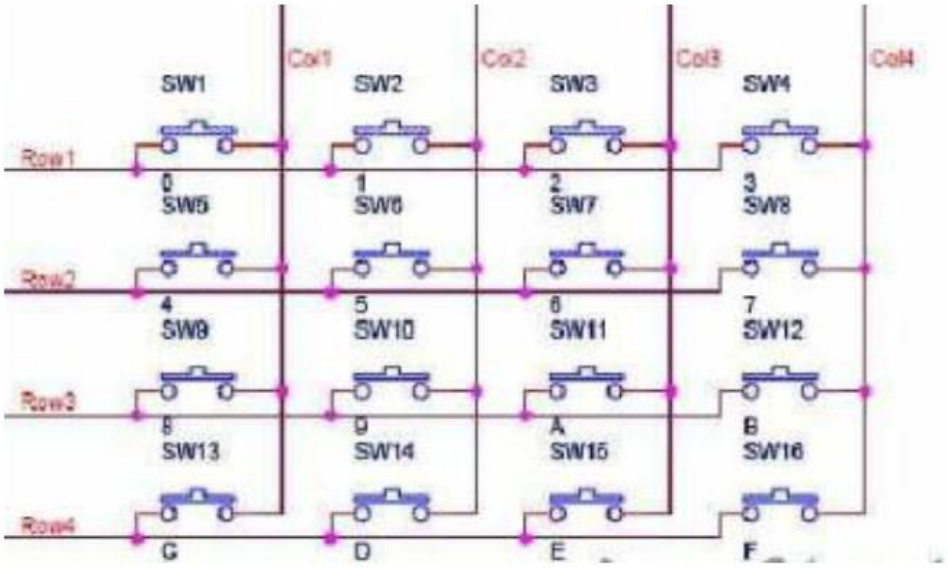
2- طريقة توصيل ال Keypad مع الميكرو كنترولر.

3- فكرة عمل ال Keypad.

4- أوامر مكتبة keypad في برنامج MicroC.

*أولا: ممَّ تتكون ال Keypad:

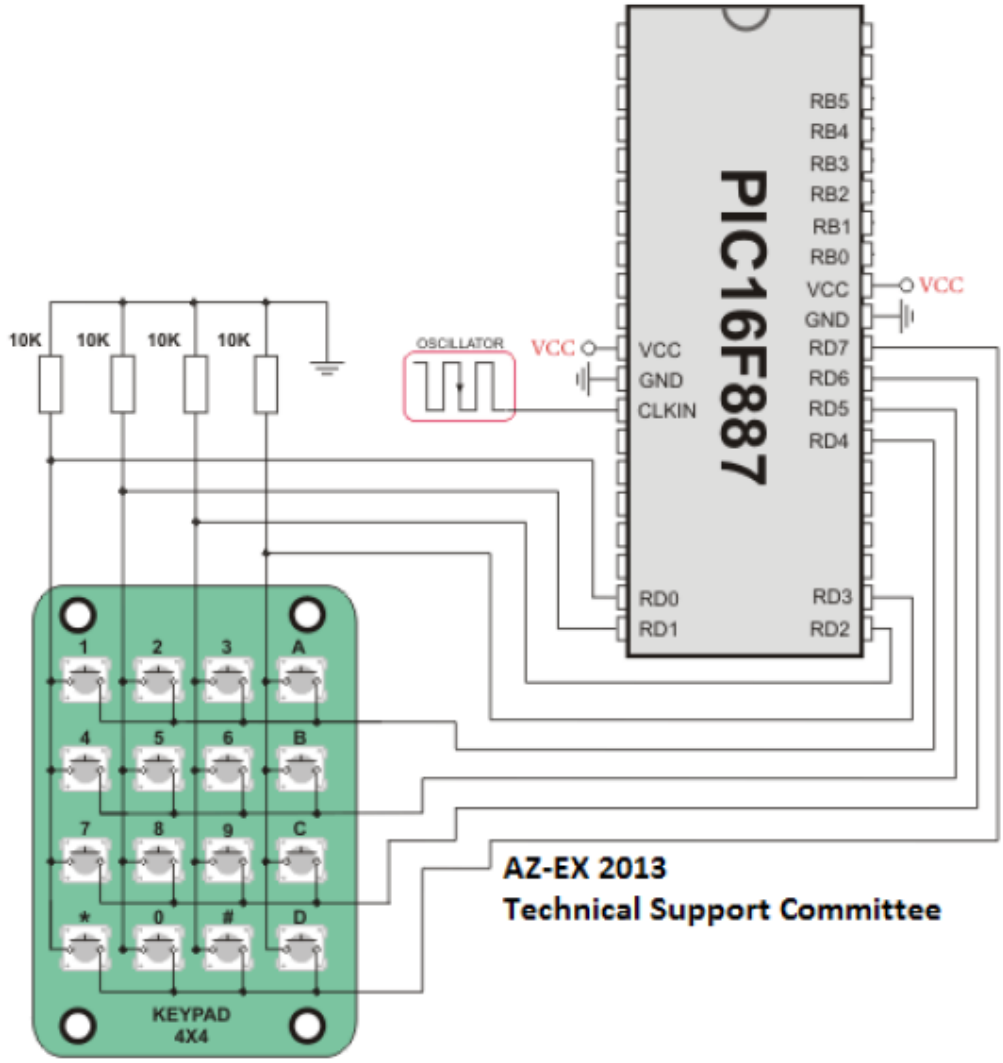
تتكون ال keypad من مجموعة من السويتشات كما هو موضح في هذا الشكل



سنجد أن لدينا ماتريكس من السويتشات بالتالي على حسب التوصيل كما سنرى وعلى حسب البرمجة فإننا نستطيع معرفة أي سويتش أو مفتاح تم الضغط عليه حيث أنه في الضغطة الواحدة يسري التيار من صف واحد إلى عمود واحد ، وهذا لكل مفتاح على حده.

***ثانيا: طريقة توصيل ال Keypad مع الميكرو كنترولر:**

طريقة توصيل ال Keypad مع الميكرو كنترولر هي في غاية البساطة ، والشكل التالي يوضح طريقة التوصيل



على سبيل المثال في الصورة السابقة تم توصيل ال Keypad على Port D ، بحيث تم توصيل الأربع أعمدة موصلة على ال 4Pins الأولى من Port D وهي من RD0

وحتى RD3 ، والأربع صفوف موصلة على ال 4Pins الأخيرة من Port D وهي من RD4 وحتى RD7.

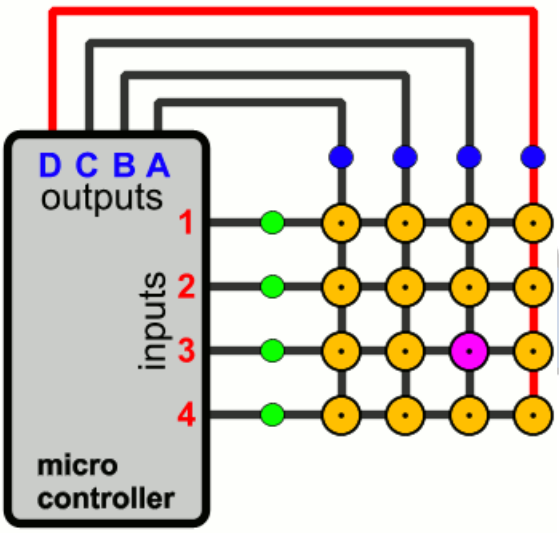
***ثالثا: فكرة عمل ال Keypad:**

يقوم الميكرو بإخراج (1) من أول 4Pins بالتتابع بحيث مثلا يخرج (1) من Portd.b0 وباقي الأربعة ستكون قيمهم ب (0) لأننا موصلين Pull down Resistance ، ثم يقوم بإخراج (1) من Portd.b1 وباقي الأربعة ستكون قيمهم ب (0) وهكذا بحيث في كل فترة من الزمن يكون عمود واحد فقط من الأعمدة عليه (1).

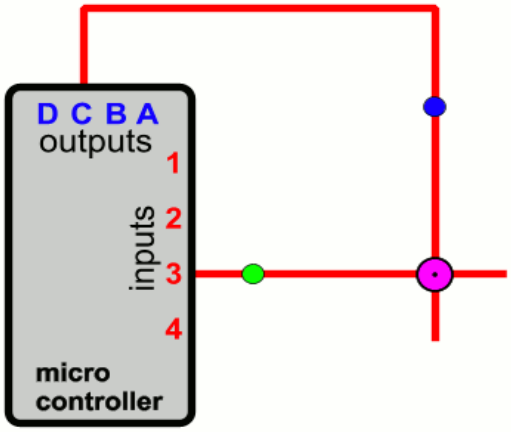
هنعتبر ان الاطراف من A/B/C/D هى ال OUTPUTS اللى بيخرج على كل منها 1 بالتبادل الاطراف من A الى D هى اطراف موصلة بالجهد الموجب

والاطرف من 1 الى 4 هى اطراف الدخل الى الميكرو والتى يقوم الميكرو باختبرها دائما لمعرفة ما اذا كان سيصلة 1 أم لا .

هذه الصورة توضيحية فقط ويجب توصيل مقومات على اطراف الدخل الى الارضى :



عند الضغط علي الزر يلمس الطرف الموجب بطرف الدخل وبذلك يصل للدخل 1 فيعرف البك أنه تم الضغط على السوتش



الخلاصة :

يتم خروج 1 من أطراف ال OUTPUTS بالتبادل ويتم ذلك بسرعة كبيرة وعن الضغط على أى سوتش يقوم بجعل طرف ال OUT يتصل مع طرف ال IN وعندما يصل لطرف ال IN واحد فأنه يقوم بتحديد مكان السوتش الذى أنضغطت علية فى أى صف وأى عمود .

كما نرى ايضا انه هناك 4 اطرف للدخل و 4 اطراف موصلة بالموجب وهناك 16 زر اى 4*4 وهذا يعنى ان هذا الكيباد 4*4

*رابعا: أوامر مكتبة keypad في برنامج MicroC:

لحسن الحظ أن برنامج MicroC موجود به مكتبة جاهزة تقوم بالعملية التي شرحناها في النقطة السابقة ، وسنقوم بشرح كل كود ووظيفته.

التحكم فى الكيباد له اكثر من طريقة
ولكنى ساشرح الطريقة الموجودة داخل مكتبات الميكروسى
وهم ثلاث اوامر

كود

```
Keypad_Init
Keypad_Key_Press
Keypad_Key_Click
```

الامر الاول

كود

```
Keypad_Init
```

وهو امر تعريف ربط اطراف الميكرو مع الكيباد

وهو يكتب داخل البرنامج اى بعد الداله الرئيسة

```
void main()
```

وتعريف الاطراف يكون هكذا

كود

```
char keypadPort at PORTD;
```

وهنا جعلت portd هو البورت الخاص بالكيباد ويمكن تغير البورت كما نريد فقط فقط نكتب بدل portd نكتب ال port الذى نريدة .

وهذا الامر يكتب فى بداية البرنامج قبل الداله الرئيسة

كود

```
void main()
```

الامر الثانى
كود

```
Keypad_Key_Press
```

هذا الكود يقوم بقراءة رقم السويتش المضغوط عليه، بمجرد الضغط عليه، وفي حالة عدم الضغط على أي سويتش يقوم بإرجاع رقم 0.
فإذا نظرنا لهذه الصورة



سنجد أن لكل سويتش رقم مقابل فمثلا إذا ضغطنا على رقم 9 يقوم هذا الكود بإرجاع رقم 11.

نجد ان هذه المكتبة لا تستطيع قراءة اكثر من 16 زر وهو لا ينتظر ان ترفع يدك من على الزر.

الامر الثالث

كود

```
Keypad_Key_Click
```

هو يقوم بنفس عمل الامر السابق (Keypad_Key_Press) ولكن الفرق بينه وبين الأمر السابق أن هذا الأمر لا يقوم بإرجاع الرقم المناظر للسويتش إلا بعد أن تزيل يدك من على السويتش، وإذا ضغطت على أكثر من سويتش في نفس الوقت لن يقوم بإرجاع أي أرقام حتى تزيل يدك عن جميع السويتشات، وبعد إزالة يدك من جميع السويتشات سيقوم بإرجاع الرقم المناظر لأول رقم ضغطت عليه.

مثال:

```
char keypadPort at PORTD;
Keypad_Init();
char kp;
kp = Keypad_Key_Click();
```

المشروع الاول

فكرة المشروع :

ربط الكيباد مع الميكرو مع ال lcd

```
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;

char  keypadPort at PORTD;
char kp;
void main() {
Lcd_Init();
Keypad_Init();
Lcd_Cmd(_LCD_CLEAR);
Lcd_Cmd(_LCD_CURSOR_OFF);
Lcd_Out(2, 5, "pressed");

while(1){
 kp=0;
while(!kp){kp = Keypad_Key_Click();}

        switch (kp) {
```

```
        case  1: kp = 49; break; // 1        // Uncomment this block for keypad4x4
        case  2: kp = 50; break; // 2
        case  3: kp = 51; break; // 3
        case  4: kp = 65; break; // A
        case  5: kp = 52; break; // 4
        case  6: kp = 53; break; // 5
        case  7: kp = 54; break; // 6
        case  8: kp = 66; break; // B
        case  9: kp = 55; break; // 7
        case 10: kp = 56; break; // 8
        case 11: kp = 57; break; // 9
        case 12: kp = 67; break; // C
        case 13: kp = 42; break; // *
        case 14: kp = 48; break; // 0
        case 15: kp = 35; break; // #
        case 16: kp = 68; break; // D

      }
  Lcd_Chr(1, 7, kp);
                        }
}
```

شرح الكود :

فى البداية قمت بتعريف الشاشة وقمت بتعريف أمر ربط أطراف الكيباد بال portd

```
char  keypadPort at PORTD
```

قمت بتعريف متغير kp لكى أحفظ فيه قيمة الزر الذى يتم الضغط عليه ثم يبدأ البرنامج فنقوم داخل main بعمل الاتى :
نقوم بعمل initialization للكيباد وللل lcd ثم نعرض على الشاشة كلمة pressed وهنا نبدأ فى البرنامج الرئيسى .

كود :

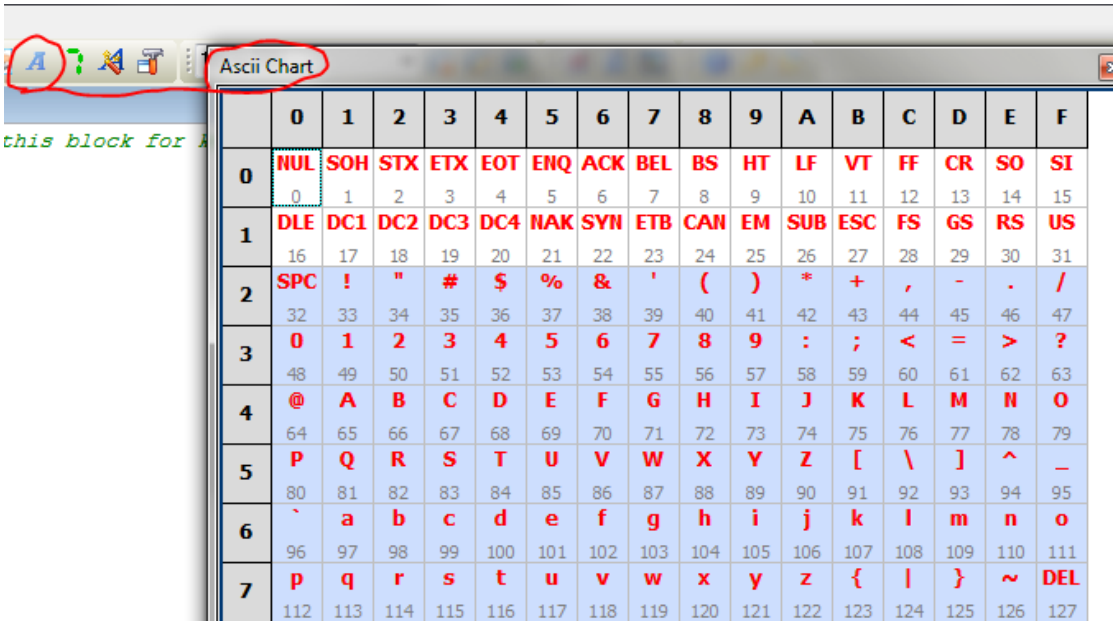
```
while(1){
 kp=0;
while(!kp){kp = Keypad_Key_Click();}
```

قمنا بعمل infinite loop وقمنا بتعريف متغير kp وأعطيناه قيمة 0
ثم قلت له طالما المتغير kp=0 لا تفعل شئ سوى أن تختبر أطراف الكيباد
هل تم الضغط على زر أم لا فأن تم الضغط على زر من أزرار الكيباد فسوف تصبح قيمة kp لاتساوى 0 وبالتالى يتحقق شرط while وتصبح قيمة المتغير kp قيمة تتراوح من 1 الى 16 وهذا الرقم يعبر عن الزر الذى تم الضغط عليه .
يقوم الامر:

```
kp = Keypad_Key_Click()
```

بقراءة قيمة الزر الذى تم الضغط عليه بالاسكى كود فاذا تم الضغط على الزر 1 فان قيمة kp=49 لان 1 يكافئ بالاسكى كود 49 وفى هذة الحالة يذهب ل case1 لان ال kp فيها قيمتها ب 49 ثم يعمل break ويخرج بره ال loop ويعرض قيمة ال kp على ال lcd عن طريق الامر lcd _char (1,7,kp); ويقوم بعرض الرقم KP بالاسكى كود زى ما هو لان الشاشة بتفهم اسكى كود فهتفهم الرقم 49 على أنه 1 وتعرض ع الشاشة 1 .

هذا هو جدول ال ascii code الموجود فى الميكرو سى :

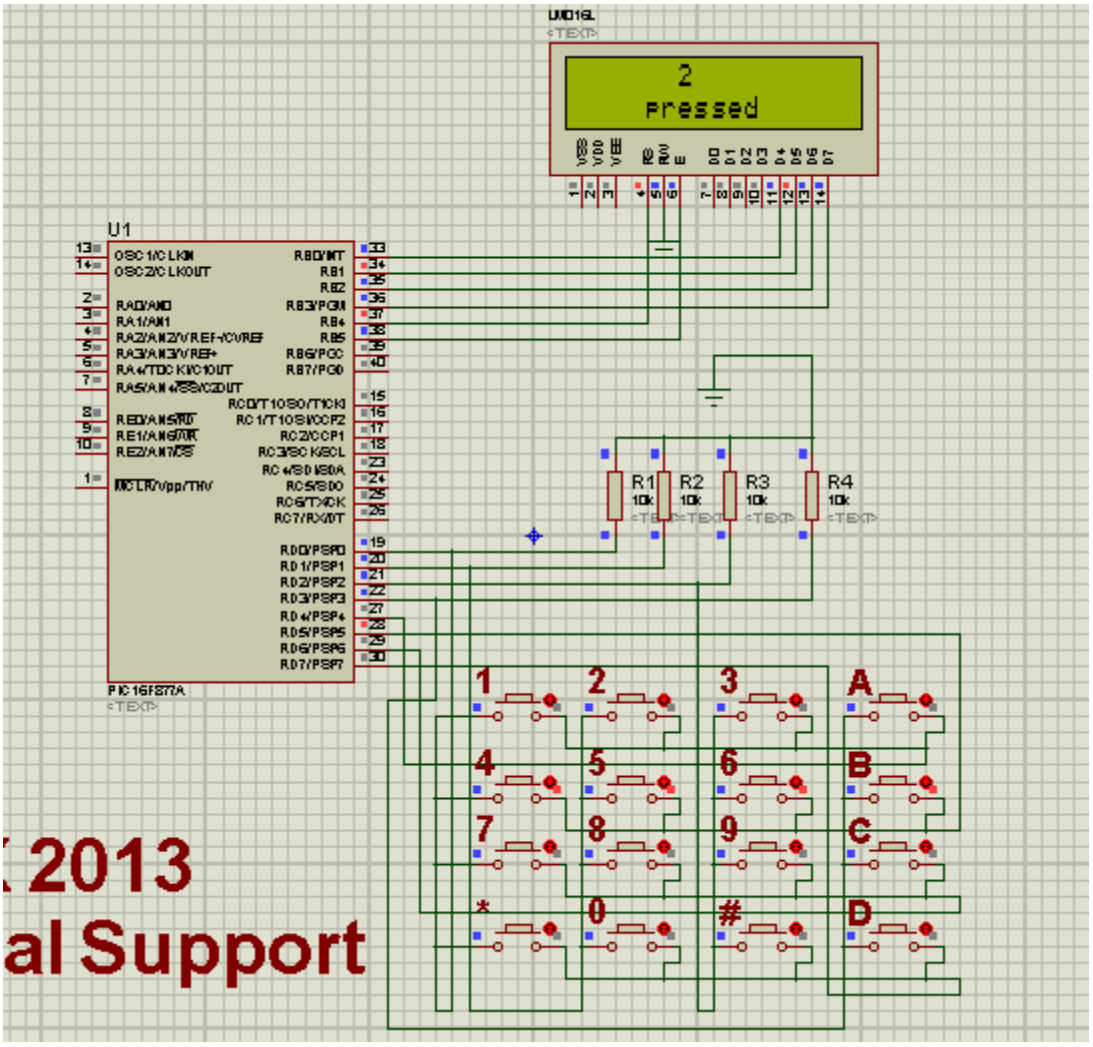


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
2	SPC 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	\| 124	} 125	~ 126	DEL 127

والذى من خلالة يتم تحديد قيمة المتغير kp والذى يكافئ قيمة السوتش الذى قمت بالضغط عليه .

```
case  1: kp = 49; break; // 1
case  2: kp = 50; break; // 2
case  3: kp = 51; break; // 3
case  4: kp = 65; break; // A
case  5: kp = 52; break; // 4
case  6: kp = 53; break; // 5
case  7: kp = 54; break; // 6
case  8: kp = 66; break; // B
case  9: kp = 55; break; // 7
case 10: kp = 56; break; // 8
case 11: kp = 57; break; // 9
case 12: kp = 67; break; // C
case 13: kp = 42; break; // *
case 14: kp = 48; break; // 0
case 15: kp = 35; break; // #
case 16: kp = 68; break; // D
```

توصيل الدائرة :



المشروع الثانى

فكرة المشروع :

في هذا المشروع يقوم الميكرو كنترولر بأخذ الرقم من المستخدم عن طريق الKeyPad ، ومن ثم يقوم بعرضها على الشاشة ويقوم بعرض عدد مرات ضغطك على السويتش أيضا.

الكود :

```
//sbit
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
//variables
unsigned short kp, cnt=0, oldstate = 0;
char txt[6];
char  keypadPort at PORTD;
//modules
void main() {
//config
//init
  Keypad_Init();                        // Initialize Keypad
  Lcd_Init();                           // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);                  // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);             // Cursor off
  Lcd_Out(1, 1, "1");
  Lcd_Out(1, 1, "Key  :");              // Write message text on LCD
  Lcd_Out(2, 1, "Times:");
//while1
while(1){
kp = 0;                                 // Reset key code variable

// Wait for key to be pressed and released
do
  // kp = Keypad_Key_Press();           // Store key code in kp variable
  kp = Keypad_Key_Click();              // Store key code in kp variable
while (!kp);
// Prepare value for output, transform key to it's ASCII value
switch (kp) {
  //case 10: kp = 42; break;  // '*'   // Uncomment this block for keypad4x3
  //case 11: kp = 48; break;  // '0'
  //case 12: kp = 35; break;  // '#'
  //default: kp += 48;
```

```
      case  1: kp = 49; break; // 1          // Uncomment this block for keypad4x4
      case  2: kp = 50; break; // 2
      case  3: kp = 51; break; // 3
      case  4: kp = 65; break; // A
      case  5: kp = 52; break; // 4
      case  6: kp = 53; break; // 5
      case  7: kp = 54; break; // 6
      case  8: kp = 66; break; // B
      case  9: kp = 55; break; // 7
      case 10: kp = 56; break; // 8
      case 11: kp = 57; break; // 9
      case 12: kp = 67; break; // C
      case 13: kp = 42; break; // *
      case 14: kp = 48; break; // 0
      case 15: kp = 35; break; // #
      case 16: kp = 68; break; // D
    }
    if (kp != oldstate) {                   // Pressed key differs from previous
      cnt = 1;
      oldstate = kp;
      }
    else {                                  // Pressed key is same as previous
      cnt++;
      }
    Lcd_Chr(1, 10, kp);                     // Print key ASCII value on LCD

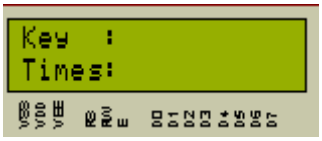
    if (cnt == 255) {                       // If counter varialble overflow
      cnt = 0;
      Lcd_Out(2, 10, "   ");
      }

    WordToStr(cnt, txt);                    // Transform counter value to string
    Lcd_Out(2, 10, txt);                    // Display counter value on LCD


}//while
}//main
```

سنجد أن الكود نفس الكود الخاص بالمشروع السابق لكن يزيد بعض الاوامر لعرض عدد مرات الضغط ع السوتش الخاص بالرقم نجد أنه فى بداية البرنامج يظهر رسالة ثابته ع الشاشة عن طريق الامرين :

```
Lcd_Out(1, 1, "Key  :");                  // Write message text on LCD
Lcd_Out(2, 1, "Times:");
```



نقوم بتعريف المتغيرات oldstate و cnt

```
unsigned short kp, cnt=0, oldstate = 0;
char txt[6];
```

اذا تم الضغط على زر رقم 1 فان قيمة kp=49 فهحتاج متغير اسمه old state لتخزين القيمة القديمة لل kp فاذا كانت أول مرة اضغطت على الزر 1 فان الشرط

```
if (kp != oldstate) {                   // Pressed key differs from previous
  cnt = 1;
  oldstate = kp;
  }
```

يتحقق لان kp ≠0 ووقتها هيجعل cnt=1 لانى لم اضغط على الزر واحد سوى مرة وسيتم عرض قيمة cnt على الشاشة من خلال الامر

```
WordToStr(cnt, txt);                    // Transform counter value to string
Lcd_Out(2, 10, txt);                    // Display counter value on LCD
```

ويتم تخزين قيمة kp فى ال old state عن طريق الامر

```
oldstate = kp;
```

فاذا تم الضغط مرة أخرى على الزر 1 فان قيمة kp=old state ووقتها لا يتحقق الشرط الاول ويتحقق شرط else

```
else {                                  // Pressed key is same as previous
  cnt++;
  }
```

وتزداد قيمة cnt وتصبح ب 2 وتعرض على الشاشة وهكذا

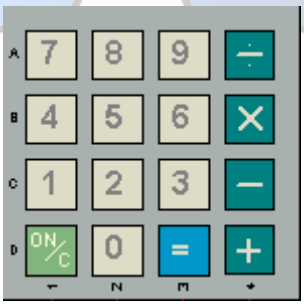
فاذا قمت بالضغط على زر 2 فان kp≠ old state وبالتالى شرط if يتحقق ويعرض ع الشاشة انى ضغطت ع السوتش مرة واحدة وبذلك أكون تمكنت من معرفة عدد المرات التى قمت بالضغط على السوتش فيها وتمكنت من عرضها ع الشاشة أيضا .

ويتم اضافة هذة الاوامر فى الكود ان لو ضغطت ع السوتش بحيث اصبح عدد مرات cnt=255 كده هيحصل over flow لانى معرفها على انها unsigned وقتها هقولة اجعل قيمة cnt=0 ويظهر عدد مرات الضغط 0 وبكده اقدر اعد عدد مرات الضغط على السوتش من أول وجديد .

```
if (cnt == 255) {                       // If counter varialble overflow
  cnt = 0;
  Lcd_Out(2, 10, "   ");
  }
```

نفس توصيل الدائرة السابقة

لا حظ هذا النوع من ال keypad :

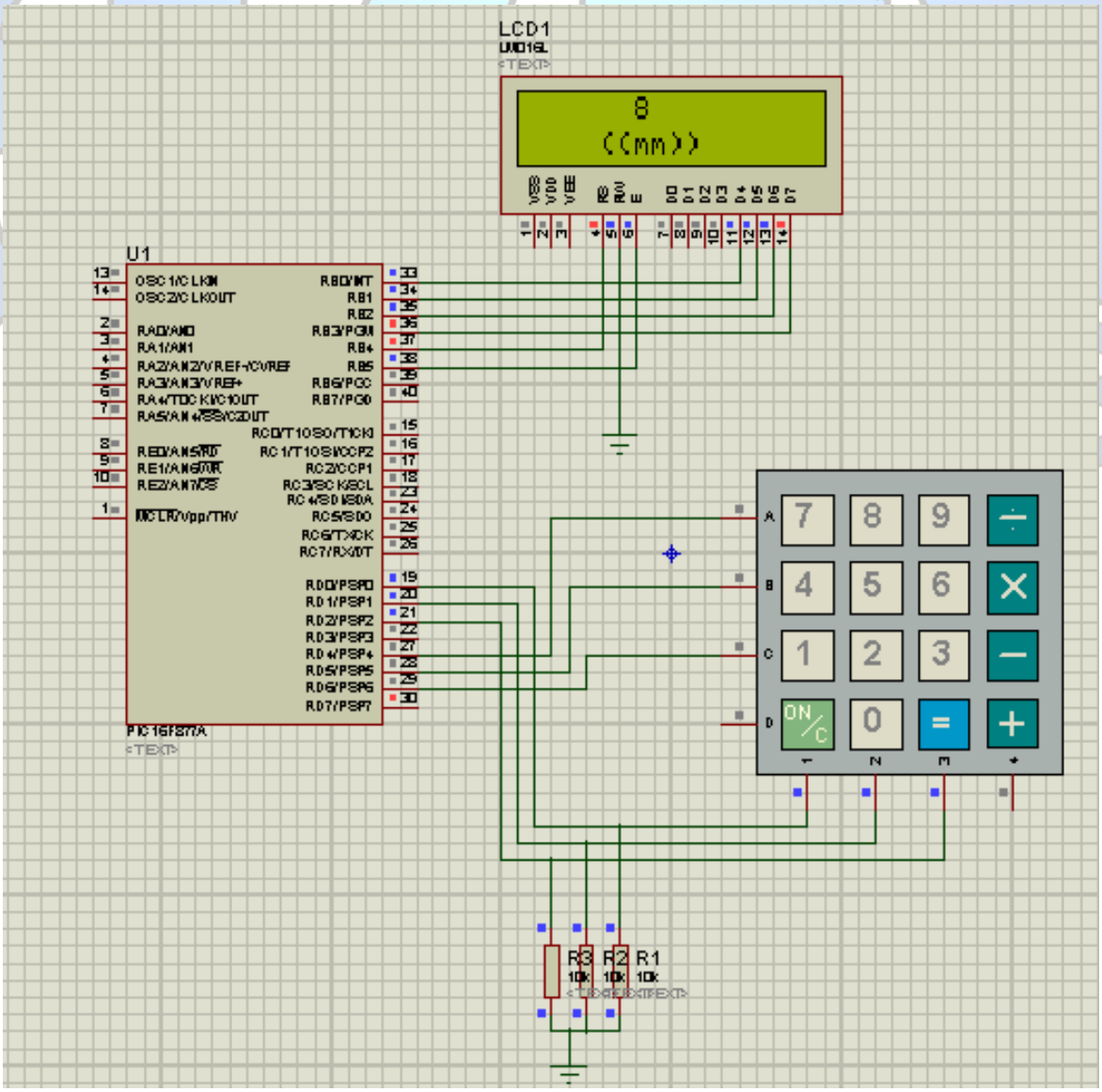


نجد أن رقم 1 يكافئ 7 و 7 تكافئ 1 لان الترتيب معكوس عن الكيباد السابقة .

```
case  1: mm = 55; break; // 1
case  2: mm = 56; break; // 2
case  3: mm = 57; break; // 3
case  4: mm = 65; break; // A
case  5: mm = 52; break; // 4
case  6: mm = 53; break; // 5
case  7: mm = 54; break; // 6
case  8: mm = 66; break; // B
case  9: mm = 49; break; // 7
case 10: mm = 50; break; // 8
case 11: mm = 51; break; // 9
case 12: mm = 67; break; // C
case 13: mm = 42; break; // *
case 14: mm = 48; break; // 0
case 15: mm = 35; break; // #
case 16: mm = 68; break; // D
```

لذا فان هذة هى ال cases الخاصة بها .

لاحظ اذا قمنا بعدم توصيل 4 line و line D فان عند الضغط على (on/0/=/×/÷/_/+) لا يظهر على الشاشة شئ .(راجع الكود الموجود فى المشاريع المرفقة مع ملف الشرح)



Chapter (10)

Stepper Motor



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

المحرك الخطوي (Stepper motor)

يستخدم في الآلات الصغيرة التي تحتاج لدقة في التحكم بمحركاتها مثل الروبوت والطابعة .

و من أهم ميزات هذا النوع من المحركات انه يمكن التحكم في سرعة دورامه وزاوية التوقف بدقة.

يستخدم هذا المحرك أيضا في التطبيقات الروبوتية، نظراً لإمكانية التحكم في إيقافه عند زاوية محددة.

ومما يميز هذا المحرك أيضا أنه يعتمد على النظام الثنائي في التشغيل حيث يلاحظ أنه يخرج منه أربع أو خمس أسلاك.

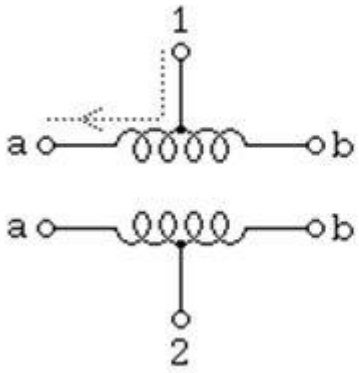
تسمح له بتلقي تتابع معين. فمثلا إذا استقبل التتابع الآتي :0001 التي تكافيء 1 في النظام العشري فإنه سيتحرك خطوة واحدة في اتجاه دوران معين أما في حال استقبال 1000 التي تكافيء 8 في النظام العشري فإنه يدور في الاتجاه المعاكس أغلب المحركات الخطوية تكون الخطوة الواحدة لها تكافيء زاوية مقدارها 1.8 درجة

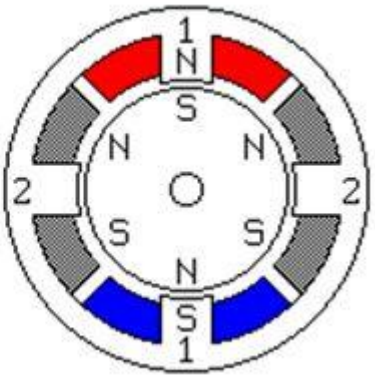
كخلاصة: إذا أردت تحريك المحرك في اتجاه معين وليكن عكس عقارب الساعة سوف ترسل له تتابع كالآتي:(0001 - 0010 - 0100 – 1000).

أرسل عكس التتابع السابق وستحصل على الاتجاه المعاكس وهذة الطريقة تعرف بأحادية القطب أو Unipolar وهذه الطريقة تستخدم عندما نريد الحصول على دقة عالية وباستخدام أقل طاقة ممكنة وهناك طريقة ثانية تسمى ثنائية القطب أو Bipolar وهذه الطريقة تستخدم للحصول على أعلى عزم ممكن من الموتور وهى كالاتى(1100-0110-0011-1001)

محرك الخطوة إحادى القطبية Unipolar stepper motor :

محرك الخطوة أحادى القطبية يكون له خمة أو ستة أسلاك و أربع ملفات (فعليا هما ملفان مقسمان بوصلة فى الوسط "المنتصف" بينهما) . وصلات المنتصف للملفات تربط معا وتستخدم فى توصيل القدرة . تسمى بالمحركات أحادية القطبية لأن القدرة دائما تمر فى اتجاه واحد وتأتى من توصيلة الوسط .

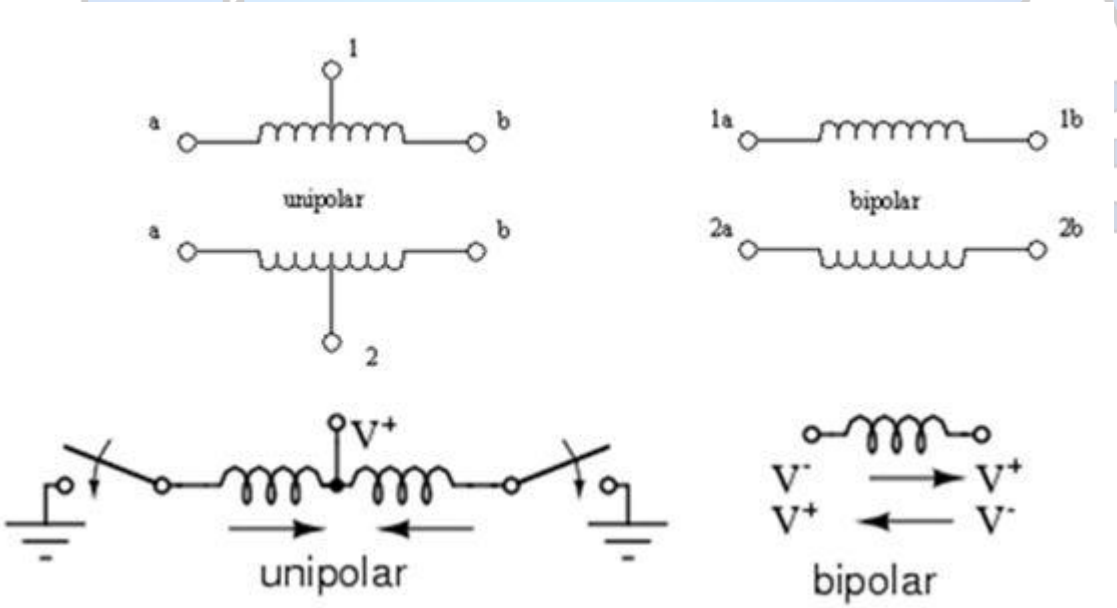




محركات الخطوة ثنائية القطبية Bipolar stepper motor :

محرك الخطوة ثنائى القطبية عادة يكون له أربعة أسلاك تخرج منه . على خلاف المحركات أوحادية القطبية , فالمحركات ثنائية القطبية ليس لها وصلة طرف أوسط مشترك , وبدلا من ذلك لها مجموعتان من الملفات المستقلة عن بعضها البعض . يمكنك التفرقة بينها وبيم المحركات أحادية القطبية عن طريق قياس المقاومة بين الأسلاك . يجب أن تجد عدد 2 زوج من الأسلاك بمقاومة متساوية .

سوف نركز على المحركات أحادية القطبية حيث أنها النوع الشائع الاستخدام والمتوفر فى الأسواق .



الأطراف

يوجد نوعان رئيسيان من محركات الخطوة:

1- المحركات ثنائية القطبيةBipolar motors
. هذا النوع له ملفان ويتم التحكم فيه عن طريق تغيير اتجاه سريان التيار خلال الملفات فى تتابع صحيح
.هذه المحركات لها أربعة أطراف فقط ولا يمكن توصيلها مع هذا المشروع.

2- المحركات أحادية القطبية Unipolar motors
.هذه المحركات لها ملفات وكل ملف له طرف أوسط center-tap والتى تعامل كأربع ملفات.

هذه المحركات يمكن أن يكون لها خمسة أو ستة أو ثمانية أسلاك . المحركات ذات الخمسة أسلاك يكون الطرفان الأوسطان متصلان ببعضهما داخليا كما فى الشكل رقم 1 .

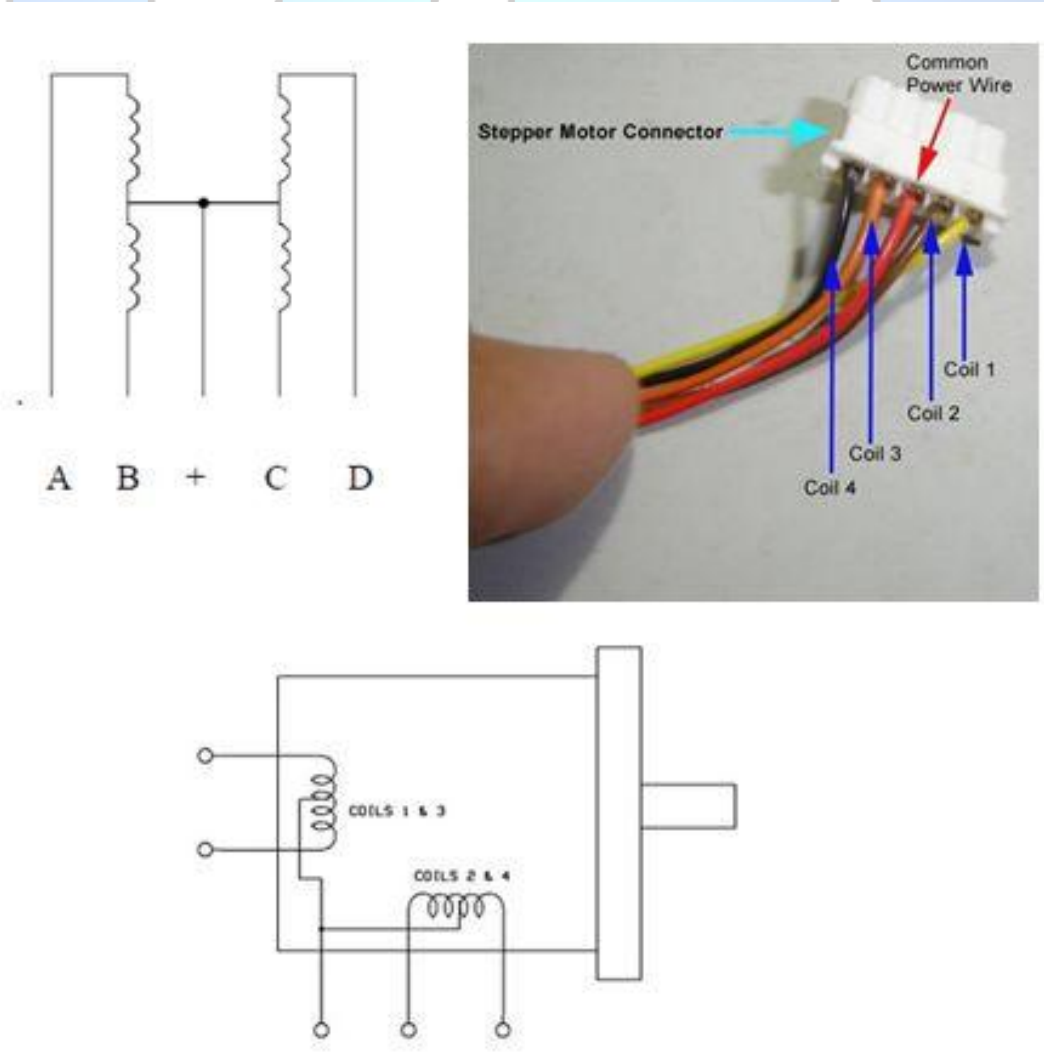


Fig 1. Five-wire stepper motor

فى المحركات ذات الستة أسلاك يتم أخراج كل طرف أوسط على حدة . نحتاج الطرفان الأوسطان لتوصيلهما معا خارجيا كما فى الشكل 2.

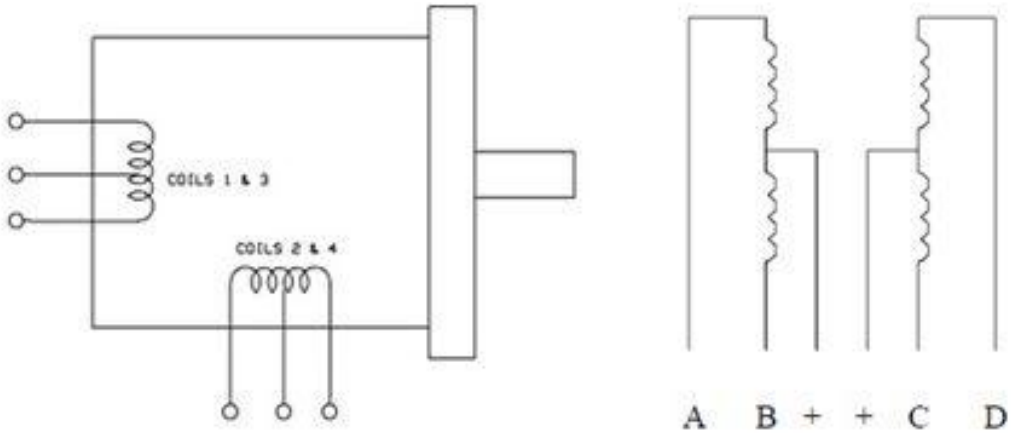


Fig 2. Six-wire stepper motor

فى المحركات ذات الثمانية أسلاك يتم إخراج نهايتى كل ملف كما فى الشكل رقم . 3 ويتم توصيل الأربعة أطراف الوسطى معا خارجيا لتشكيل سلك واحد .

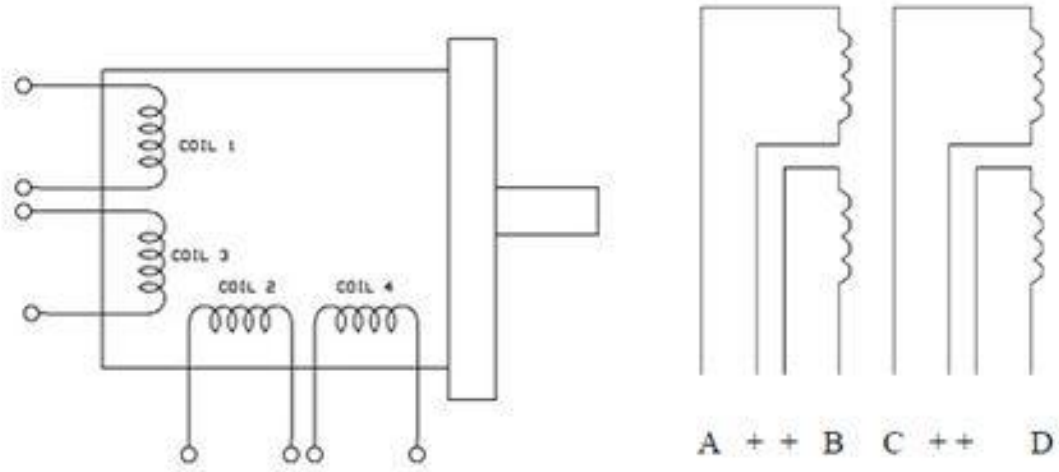


Fig 3. Eight-wire stepper motor

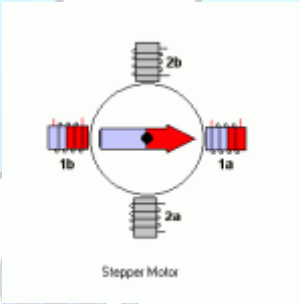
فى كل حالة يتم توصيل الطرف (أو الأطراف) الأوسط بالجهد الموجب لمصدر تغذية المحرك . قد يستخدم المحرك أحادى القطبية كمحرك ثمائى القطبية عن طريق عدم استخدام الأسلاك الموجبة.

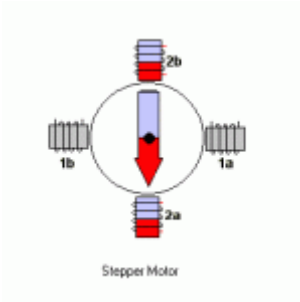
الخطوات

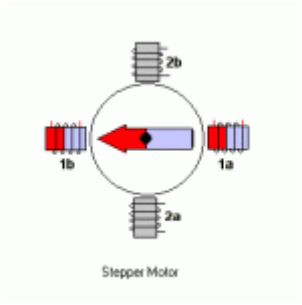
تتعدد التصميمات للمحرك الخطوى وتتعدد الملفات من نوع لآخر وتختلف طريقة توزيع الملفات كلما كانت أقطاب المحرك وعدد الملفات أكثر كلما كان أقوى عزما ولذلك نجد أن المحركات خطوية غالبا لا تكون كثيرة الأقطاب إذا قارناها مع المحركات الأخرى ويعود هذا لتلافى البرمجيات المعقدة وللسرعة الأعلى فى الحركة.

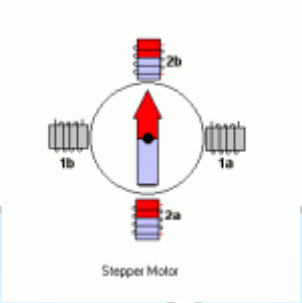
وكلمة خطوى أى إعتماده كليا فى كل خطوة على النبضات ولا يتحرك بالتيار المستمر

وهذه الصور لأربع خطوات باستخدام أربع أقطاب ونرى فيها أن كل خطوة للجسم الدوار تعتمد على تمرير تيار فى ملفان متقابلان فيعملان على جذب لأقطاب العضو الدوار (الروتور) المخالفة وتنافر الأقطاب المتشابهه وقد ميزنا كل قطب بلون مختلف بغض النظر عن تسمية الأقطاب بالشمالى والجنوبى S , N



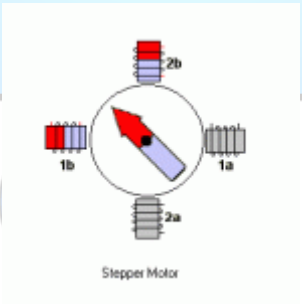


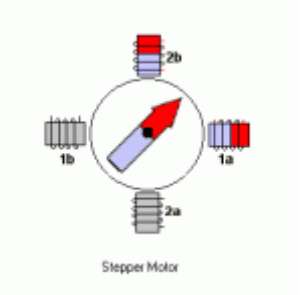


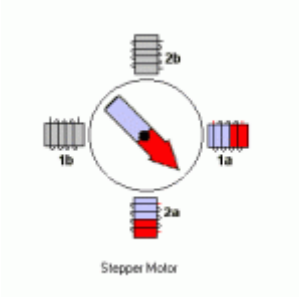


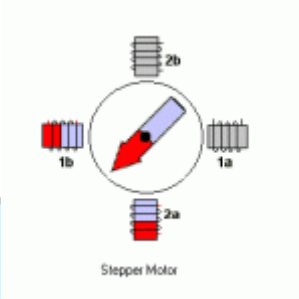
==========

ونفس الفكرة وأيضا أربع اتجاهات ولكن قوة وعزم أكبر لإستخدامنا ملفان متوازيان فى كل خطوة كما يلى :-









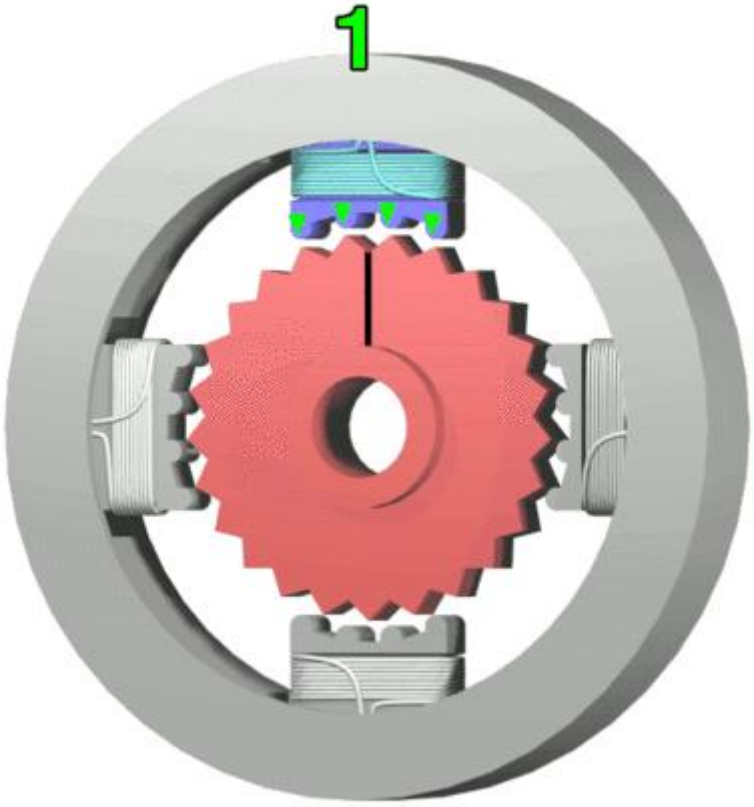
==========

ونلاحظ فى كلا الطريقتان أن العضو الدوار تحرك بزاوية 90 درجة فى كل خطوة وهناك طريقة تجمع بين الطريقتان وهى الطريقة الأولى تعمل على إعطاء نبضات على قطبى متقابلين ثم الخطوة التالية يكون أحد هذان الملفان مع ملف مجاور فى آن واحد كما فى الطريقة الثانية التى تستخدم ملفان متجاوران ، وبذلك سيكون خطوة ملفان متقابلان وخطوة أحدهما مع ملف مجاور كما فى الطريقة الثانية فيدور الغضو الدوار بزاوية 45 درجة بدلا من 90 درجة فى كل خطوة وبالتالى يكون العزم أقوى وها قد شرحنا المطلوب بغض النظر عن نوعية المواتير الخطوية التى تختلف عن بعضها بعدد أقطابها وطريقة توصيل الملفات فى داخل الموتور وعلى أى حال فها قد عرفنا طريقتان الخطوات الأساسية وبناءا على طريقة التوصيل الداخلية سنقوم بعمل البرمجة .

ويجب أن أضيف لحضراتكم أن فى الطريقة الأولى نرى أنى قمت بعمل تمرير للتيار على كل ملفان متقابلان وهذا من أجل العزم وكان يمكن أن نعمل على تمرير التيار فى ملف واحد فقط فى كل خطوة وهذا ما يستخدم فى بعض الأنواع الأخرى ولكن تكون أضعف عزما .

ونلاحظ أنه لا خلاف بين الخطوة خطوة والنصف خطوة نصف خطوة فى تركيبة الملفات ولكن الفارق هو نظام إعداد البرمجة التى تؤقت زمن دخول النبضات على الملفات ، ومن سياق الشرح نتبين أن عكس ترتيب النبضات فى أى وقت نشاء سيعمل على عكس الحركة فى الحال طالما الحركة تعتمد فى خطواتها على النبضات ، وبسرعة النبضات تكون سرعة الحركة التى هى تعنى سرعة الموتور.

والآن عرفنا كيف يتم تقسيم حركة المحرك الخطوي إلى زوايا (90 درجة ومضاعفاتها) ولكن كيف تكون الخطوة الواحدة للمحرك الخطوي بقيم صغيرة (1.8) مثلاً ؟!
شاهد هذه الصورة المتحركة للمزيد من التوضيح

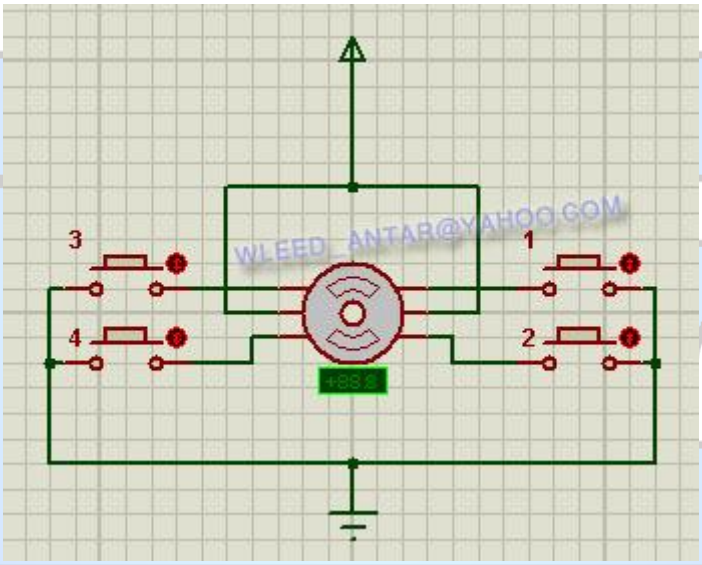


http://upload.wikimedia.org/wikipedia/commons/6/67/StepperMotor.gif

التشغيل

والآن اخترت لكم محرك خطوى هو فى الواقع عبارة عن ملفان لكل منهما نقطة منتصف مستقلة وبهذا يكون لدينا 6 أطراف نقوم بإعطاء الطرف الأوسط لهما موجب ثابت ونتحكم بنبضات سالبة على باقى الأطراف أو العكس تماما نعطى الطرفان الوسطان سالب ونتحكم بنبضات موجبة ، لن تفرق معنا.

وفى هذه الرسم سنتحكم يدويا لنرى عمل الخطوات قبل الشروع فى البرمجة :-



1- نضغط على المفتاح رقم 1 وننظر كيف اتجهت أقطاب العضو الدوار.

2- نضغط على المفتاح رقم 2 وننظر كيف اتجهت أقطاب العضو الدوار.

3- نضغط على المفتاح رقم 4 وليس 3 وننظر كيف يسير القطب فى إتجاه منتظم ولكن نلاحظ أن فى كل خطوة يقفز العضو الدوار 90 درجة ويظهر لنا هذا على التدريج.

والآن نقوم بتجربة نصف خطوات:

1- نضغط على النقطة الحمراء للمفتاح رقم 1 لتثبيت المفتاح وننظر كيف اتجه القطب.

2- نضغط على النقطة الحمراء للمفتاح رقم 2 لتثبيته وننظر كيف تحرك القطب ولكن نصف خطوة أى 45 درجة فقط وليس كما فى التجربة الأولى والسبب أن الملفان المتجاوران يعملان فى آن واحد وبنفس القطبية للمجال المغناطيسى المتولد منهما وبالتالى أصبحت قوة الجذب ليست من ملف واحد ولكن من الملفان المتجاوران وبتساوى فى القوة وبالتالى فإن القطب الدوار لن ينحاز لإحداهما ولكن يقف بينهما لا يتجه كليا للملف الأول ولا يتجه كليا للملف الثانى وبذلك أصبحت الخطوة منتصفة بين الملفان .

3- نضغط على النقطة الحمراء للمفتاح رقم 1 لتحريره وهنا يفقد الملف الأول المغنطة الكهربائية المؤقة بفعل التيار ويصبح الملف الثانى له القوة المؤثرة وحده وهنا لا يوجد سبب يجعل القطب الدوار ثابت بين الملفان ولكن يتجه كليا للملف الثانى بحركة 45 درجة وتتصف بنصف خطوة.

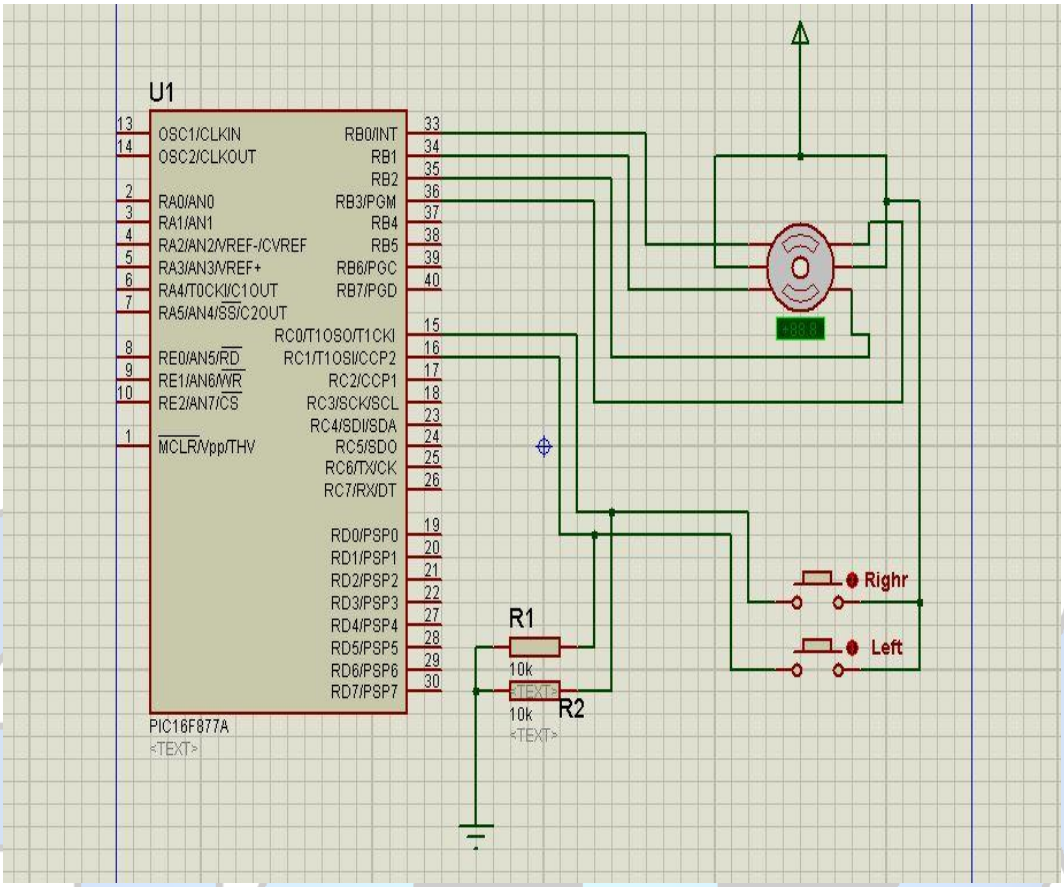
4- والآن الملف التالى فى الترتيب هو ذى الطرف المتصل بالمفتاح 4 فنقوم بالضغط على النقطة الحمراء لهذا المفتاح فنجد القطب الدوار المتجه للملف الثانى تحرك نصف خطوة وبمقدار 45 درجة بفعل المجال المتولد من الملف الثالث ذى المفتاح 4 ومادام الملف الثانى ذى المفتاح 2 لازال متصل بالتيار فإن قوة الجذب للملفان متساوية تسبب مناصفة الجذب للقطب الدوار فيقف بينهما حتى نقوم بتحرير المفتاح رقم 2 وبعدها تصبح القوة الوحيدة للجذب هى للملف الثالث فتجذب القطب تجاهها مسببه الحركة 45 درجة .

وبنفس الترتيب ولكن مع عكس المفاتيح بدلا من تفعيل الملف الأول ثم الثانى ثم الثالث نقوم بعمل تفعيل الملف الثالث ثم الثانى ثم الأول فيتجه القطب مسببا الحركة العكسية .

التشغيل بالميكروكنترولر

بالنسبة للبرمجة فهى تختلف فى صيغتها حسب سير المشروع وتفرعاته فيمكن وضع برنامج فرعى خاص بهذه الجزئية ويمكن تصميم برنامج تلقائى التصرف خاص بالروبوتات يتفرع مع التعليمات حسب المدخلات اليدوية أو التلقائية إذا كان المشروع يدعم مستقبل الآشعة تحت الحمراء أو أى شئ آخر كصناعة روبوت يتحرك إلى أن يقترب من حائط وقبل أن يصطدم به على مسافة معينة يقف ولعل الكثير رأى مثل هذه المشاريع ولكننا هنا نتناول الموضوع بعيدا عن هذه التفرعات أو المقاطعات وإنما من باب التعامل مع المحرك الخطوى عن طريق المتحكم

هذا مشروع للتنفيذ على المحاكى ولكن فى الحقيقة نحتاج لبعض الإضافات وهى ترانزستور يعمل على تشغيل الملف وبالتالى نحتاج لأربع ترانزستورات



البرنامج

```
void main()
{
portb = 0;
trisb = 0;
trisc =11111111;
while(1(
{
if(portc.f0==1)
{
```

```
portb = 0b00000011;
delay_ms(150);
portb = 0b00000001;
delay_ms(150);
portb = 0b00001001;
delay_ms(150);
portb = 0b00001000;
delay_ms(150);
portb = 0b00001100;
delay_ms(150);
portb = 0b00000100;
delay_ms(150);
portb = 0b00000110;
delay_ms(150);
portb = 0b00000010;
delay_ms(150);
}
if(portc.f1==1)
{
portb = 0b00000011;
delay_ms(200);
portb = 0b00000110;
delay_ms(200);
portb = 0b00001100;
delay_ms(200);
```

```
portb = 0b00001001;
delay_ms(200);
}
}
}
```

بعد تحميل الملف المرفق وتشغيله نلاحظ شئ !

أنه بمجرد الضغط على المفتاح right فإن الموتور يدور تجاه اليمين بحركة منتظمة سلسة .

أما فى حالة الضغط على المفتاح lift يبدأ الموتور فى الدوران جهة اليسار ولكن نلاحظ أن الدوران جهة اليسار هنا لن يكون فى حركة سلسة بل فى حركة تكاد تشبه القفز فما تفسير ذلك ؟؟

تعمدت أن أجعل الحركة جهة اليمين تتحرك بنظام النصف خطوة فى كل مرة ، وأما الحركة جهة اليسار فهى تتحرك فى كل مرة خطوة كاملة (أى أن القطب لا يستقر بين ملفان إنما ملف ملف) وقد عمدت لهذا لتوضيح الفارق بين النصف خطوة والخطوة ،

ملاحظة :

لا يهم بأى حركة نبدأ ولكن يهمنا الترتيب ، وأما عكس هذا الترتيب يسبب الحركة العكسية جهة اليسار ،

نلاحظ أيضا أن الأسطر التى تحتوى على 1 فقط هى لتشغيل ملف مفرد وأما الأسطر التى تحتوى على رقمان مثل 0011 أو 1001 فهى تعنى تشغيل ملفان إثنان ، وكما سبق وشرحنا أن تشغيل ملفان أقوى عزما ، وأما تشغيل ملف ثم ملفان فهذا يعنى الحركة نصف خطوة كل مرة وبزاوية 45 درجة وأما الكود الخاص بالحركة اليسرى فلا يحتوى إلا على أسطر تعمل على تشغيل ملفان فى كل مرة وبذلك يتم التحرك خطوة كاملة فى كل مرة والآن تعاملنا مع المحرك الخطوي وعرفنا كيفية تشغيله والتعامل معه، ولكن إذا كنّا نستخدم المحرك الخطوي عملياً لدقته في الحركة ولإمكانية عمل زوايا معينة به، فكيف يمكن عمل هذه الزوايا برمجياً ؟!

مثال :
إذا كان لدينا محرك خطوي خطوته 1 درجة مئوية ونريد أن نحرك الموتور بزاوية مقدارها 35 درجة

أولاً: يمكن عمل ذلك بطريقة تقليدية وذلك عن طريق تعريف البورت المتصل عليه المحرك 35 مرة كل مرة يتمّ إخراج 1 على أحد الأطرف بالترتيب وبذلك يتحرك المحرك 35 خطوة أي يتحرك بزاوية مقدارها 35 درجة ولكن هذه الطريقة ليست عملية ولن تكون مناسبة مع الزوايا الأكبر لذلك فإننا نستخدم طريقة برمجية أسهل فبدلاً من تكرار تعريف البورت يتم تعريفه مرة واحدة واستخدام for loop لكي تقوم بتكرار الأمر حسب العدد المحدد
بداية نقوم بتعريف البورت وليكن كالتالي :

```
portb=0b10001000;
```

وهكذا قمنا بتعيين 1 على بين واحدة فقط متصل بالمحرك وبين أخرى غير متصلة بالمحرك وبالتالي تحريك المحرك خطوة واحدة فقط، ونريد تعيين بقية الأربع بينات المتصلة بالموتور ب1 على التوالي، فبدلاً من تعيين البين كل مرة نستخدم أمر برمجي لتسهيل تلك العملية وهو :

```
portb=(portb<<1)|(portb>>7);
```

لنفهم الأمر يجب أن نفهم كل جزء فيه
أولاً portb<<1 يقوم بتحريك القيم المخزنة في البورت حركة واحدة إلى اليسار فيصبح البورت كالتالي portb=0b0001000
ثانياً portb>>7 يقوم بتحريك القيم المخزنة في البورت 7 حركات إلى اليمين فيصبح البورت كالتالي portb=0b00000001
ثم نقوم بجمع (OR) قيمة البورت الأولى مع قيمة البورت الثانية وتخزينهم في نفس البورت فيصبح الناتج portb=0b00010001
وإذا لاحظت فإننا بهذا الأمر قمنا بنقل الـ1 إلى البن التالية (الملف التالي في المحرك)
وبالتالي تحريك المحرك خطوة أخرى بالترتيب في اتجاه معين (إذا أردنا عكس الاتجاهه نقوم بعكس اتجاه الأسهم >> <<)
إذن فهذا الأمر يقوم بنقل الواحد على البورت بالترتيب المطلوب وما علينا إلا تكرار هذا الأمر بعدد الخطوات الذي نريده وبالتالي تحريك المحرك بمقدار الزاوية التي نريدها فنستخدم هذا الأمر داخل for loop ونقوم بتحديد عدد مرات تنفيذ اللوب بنفس عدد الخطوات التي نريدها فيكون الكود كالتالي

for loop.c

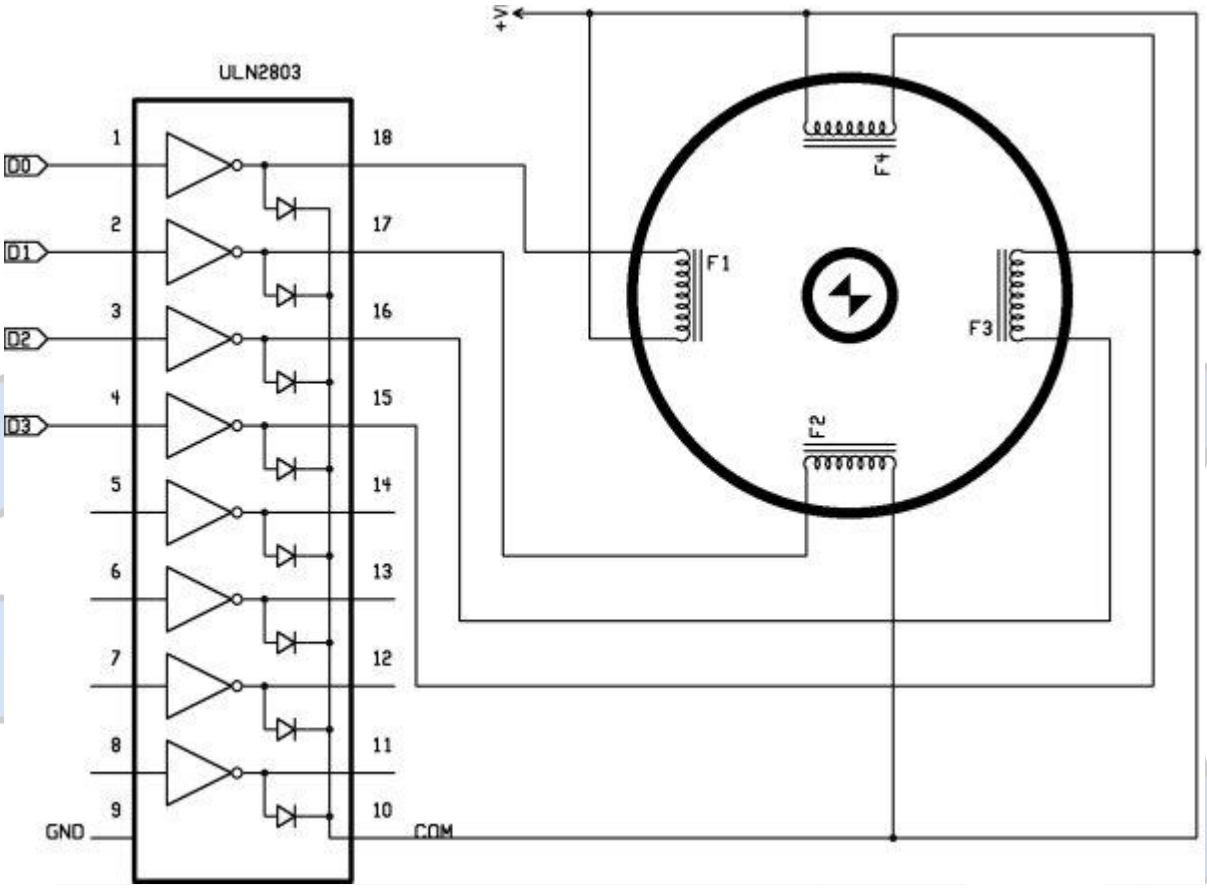
```
void main() {
int i=0;
trisb=0;
trisc=255;
portb=0b10001000;

while(1){
if(portc.f0==1){
for(i=0;i<35;i++)
{
portb=(portb<<1)|(portb>>7);
delay_ms(100);
}
}
if(portc.f1==1){
for(i=0;i<35;i++)
{
portb=(portb>>1)|(portb<<7);
delay_ms(100);
}
}
}
```

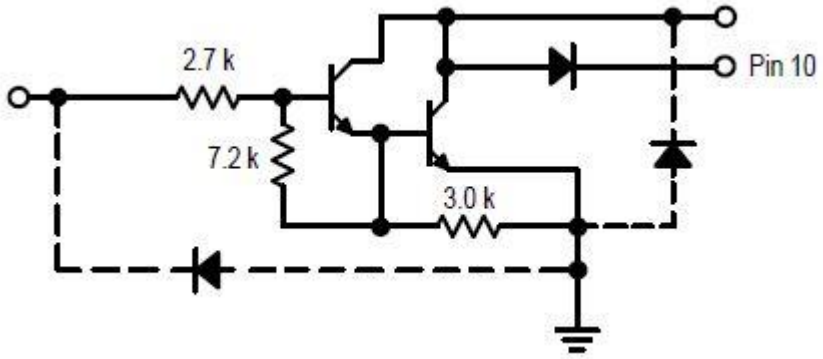
راجع ملف البرنامج والمحاكاة في المرفقات

ملحوظة مهمة جداً:

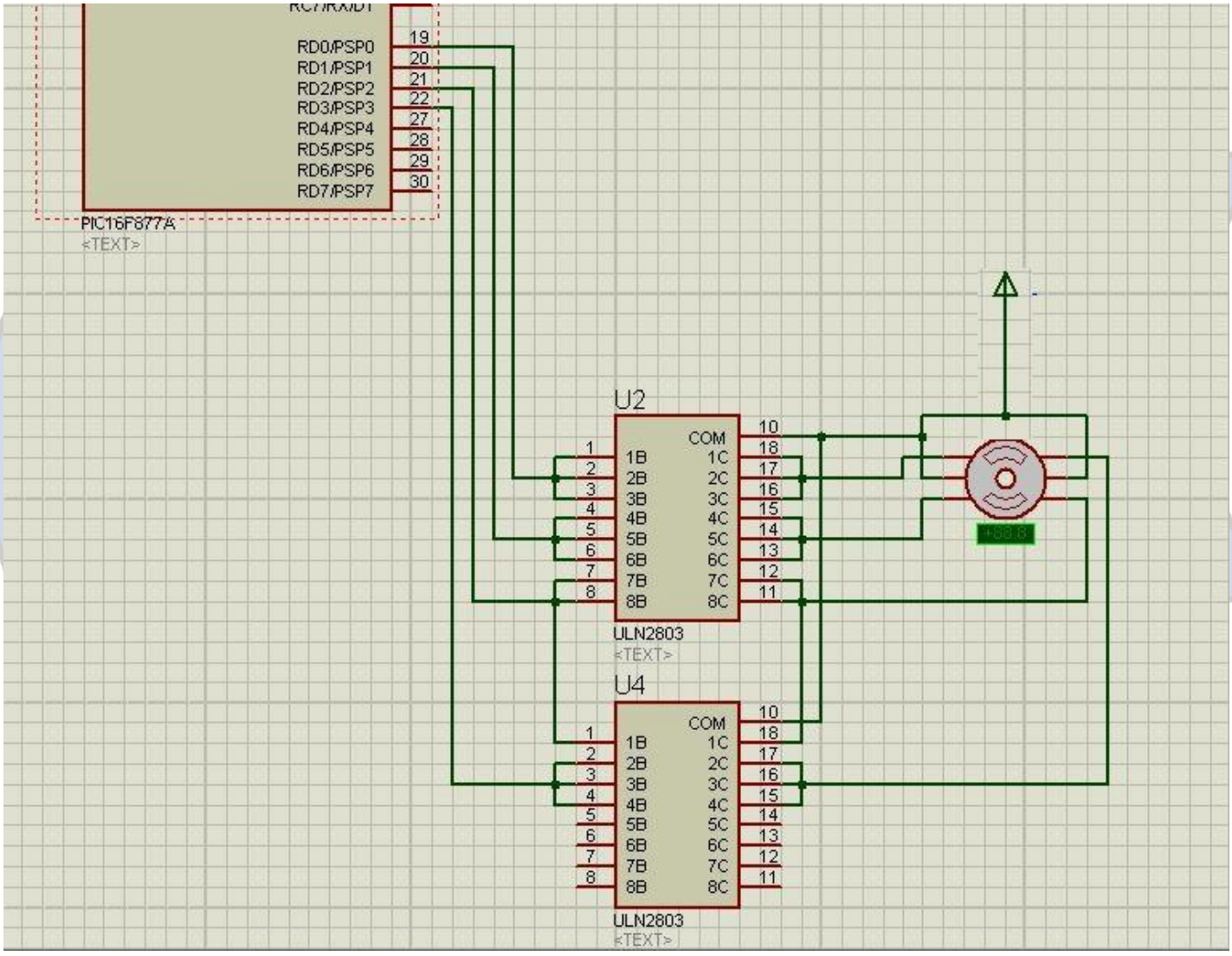
عند التوصيل عملياً يجب العزل بين المحرك الخطوي (stepper motor) وبين البيك ميكروكنرولر، وعادة ما نستخدم للعزل IC ULN2803 حيث تتكون من مجموعة من الترانزستور من النوع دارلنجتون ومجموعة من الدايود FREEWHEELING DIODES



وهذه هي التوصيلة الداخلية لكل رجل pin من رجول الدائرة المتكاملة



تتحمل البين للدائرة المتكاملة تقريباً 500 مللي أمبير وحتى 50 فولت، لذلك عندما نستخدم موتور يستهلك تيار أكبر من 500 مللي أمبير نستخدم أكثر من بين من الدائرة المتكاملة لكل طرف من الموتور، فإذا كان الموتور يستهلك 1.5 أمبير مثلاً، نستخدم لكل طرف 3 بين عن طريق ربطهم سوياً لكل بين 500 مللي أمبير، لذلك قد نستخدم أكثر من دائرة متكاملة للموتور الواحد كما بالشكل التالي



ملحوظة :

تعمل IC ULN2803 كعاكس للجهد Not gate فإذا كان الخرج من البيك 1 تجعله الآي سي 0 والعكس صحيح، فعمليا يكون الموتور متصل بمصدر التغذية ولكن الطرف الآخر من الملف غير متصل بالأرضي وعندما يتصل ب 0 أو الأرضي عن طريق الآي سي (1 في كود البرمجة) يمر التيار في الملف فيمغنطه مما يسبب دوران الموتور خطوة واحدة وهكذا مع بقية الملفات .

Chapter (11)

Serial Communication



www.az-ex.net
twitter.com/azex2013
facebook.com/alazharexhibition
facebook.com/groups/azex.2013

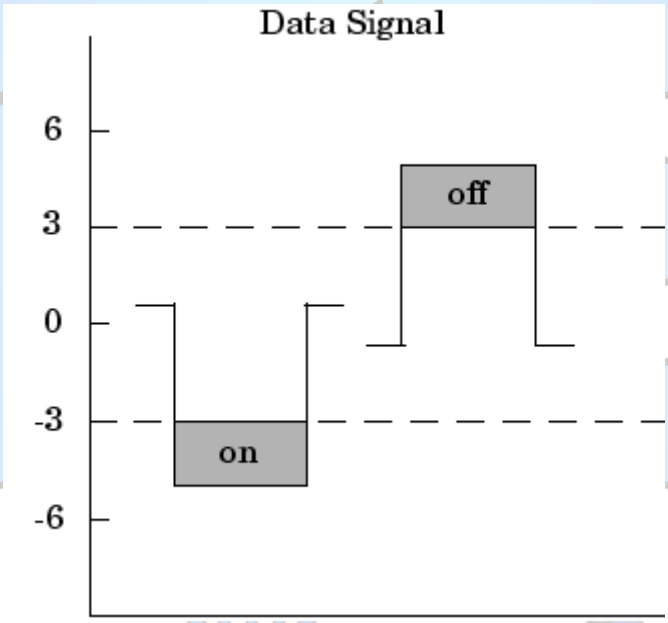
Serial Port

هو عبارة عن منفذ بين الكمبيوتر و الأجهزة الأخرى و تمتلك بروتوكول خاص بها سوف نتطرق له إن شاء الله هذا البروتوكول يستخدم لربط 2 devices أو أكثر الاول يكون الكومبيوتر والثانى من الممكان أن يكون طابعة أو الميكرو ومعنى كلمة بروتوكول مجموعة من القواعد التى تحكم أمر معين وهو الاتصال بين الكومبيوتر واى جهاز اخر هذة القواعد منها :

1_ أن الداتا بتتبعت serial واحدة تلو الاخرى .

2_ ال logic level للسريال فنجد أن ال high level للسريال لما يكون الجهد أكبر من 3- بالسالب يعنى من -4 الى -25

أما ال low level لما يكون الجهد أكبر من 3 بالموجب يعنى من 4 الى 25 .



وسنذكر هذة القواعد لاحقا .

باستخدم هذا الكابل لتوصيل ال 2 devices ببعض :



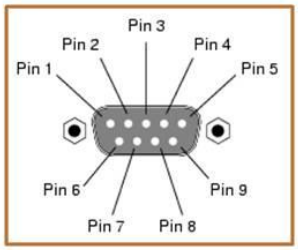
البيانات يتم ارسلها عن طريق هذا الكابل serial واحدة تلو الاخرى bit by bit عشان كده مش بحتاج غير line واحد أما لو هرسل واستقبل فى نفس الوقت هحتاج 2 lines ويسمى

فى هذة الحالة directional وكده لابد من سلكين full-duplex أما اذا تم الارسال والاستقبال من نفس السلك فانه لا يكون فى نفس الوقت ويسمى half-duplex .

للسريال منفذان أحدهما male والاخر female و تسمى منافذها بال COM ports .

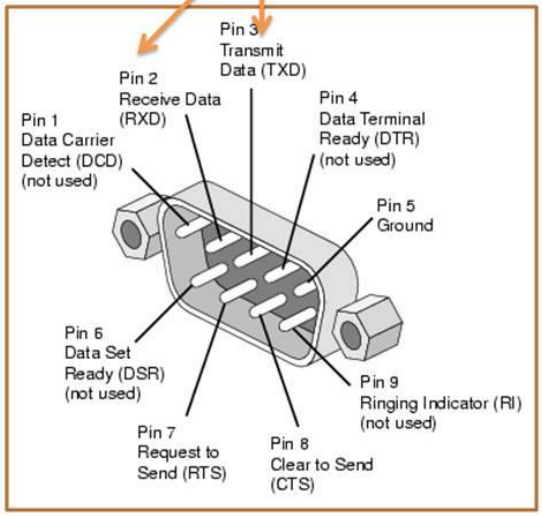


Pin assignment



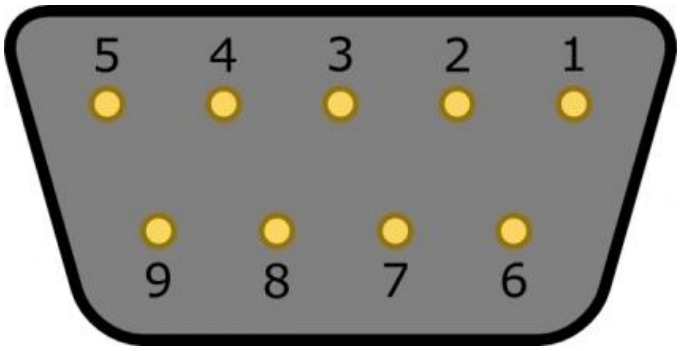
Pin	Assignment	Description
1	DCD	Data carrier detect
2	RXD	Receive data
3	TXD	Transmit data
4	DTR	Data terminal ready
5	GND	Signal ground
6	DSR	Data set ready
7	RTS	Request to send
8	CTS	Clear to send
9	RI	Ring indicator

الارجل التى نهتم بها هى
الارجل 2 و 3
أى الاستقبال والارسال



محتاج أعرف 3 pins هما واحد بيرسل pin3 واحد بيستقبل pin2 والارضى pin5

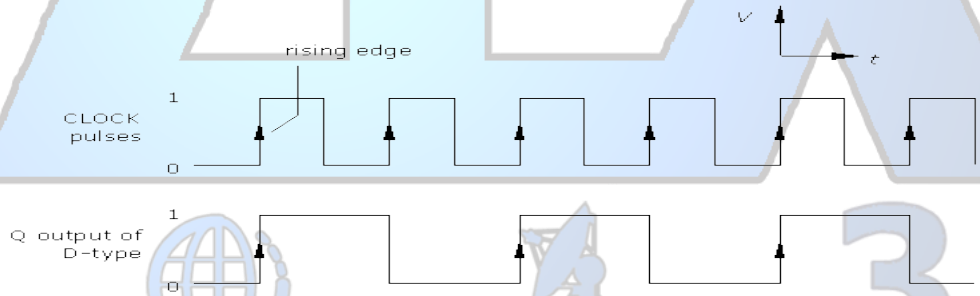
الترقيم دا خاص بال male أما ال female ترقيمها عكس ال male عشان لما يتركبوا فى بعض ودا شكل ال female .



Synchronous and asynchronous communication

Synchronous communication:

يعنى تزامنى بمعنى انه كل لحظة زمنية محددة وبقيمة معينة هيقوم بوظيفة معينة فمثلا لو عندى دائرة كونتر نتعد من 0الى 9 بين كل عدة والتانيه نص ثانية ويتم تحديد الزمن عن طريق ال clock



الكومبيوتر هيقول للميكرو هيوصلك أمر كل نص ثانية فالميكرو هيجهز نفسه انه هيستقبل أمر كل نص ثانية .

Asynchronous communication:

مش مرتبط فى الارسال والاستقبال بزمن معين لكن مرتبط بعلامة معينة الميكرو يقول للميكرو ضع ع ال PIN اللى هتستقبل عليه 1 عشان ارسلك البيانات وساعة ما الكومبيوتر يبعت للميكرو 0 يعرف أن اللى بعد ال 0 هى الداتا اللى هيستقبلها .

Serial data format

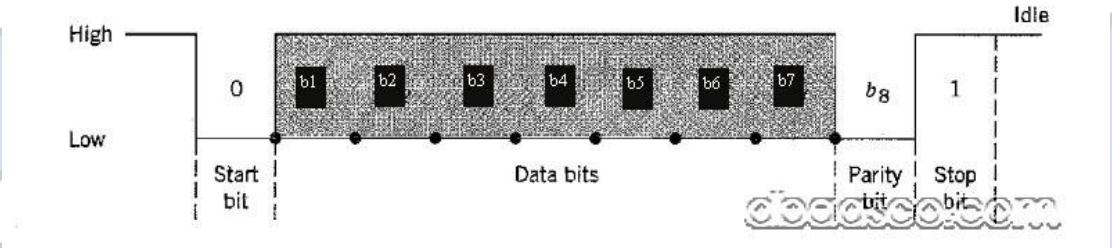
احنا بنشتغل asynchronous الداتا هيكون ليها format معين هتتبعت بيه هتتبعت الداتا ومعاها other bits وهى :

1_Start bit

2_data bits

3_parity bit

4_ Stop bits

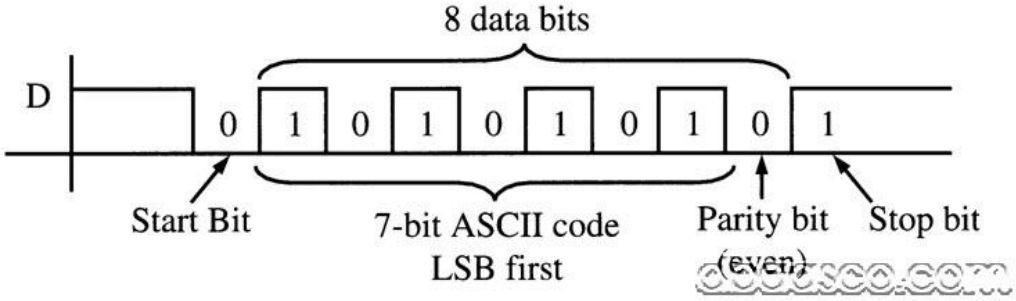


أول حاجة هتتبعت من الداتا هو ال start bit وهو الذى يحدد بداية عملية الارسال لو ال start bit يساوى zero فا الميكرو هيستقبل الداتا اللى بعده طبعا بعد ال start bit هيكون ال data bits وهى البيانات المرسلة بال ASCII Code وعددها 8 بيت بعد ال data bits .

هيكون ال parity bit وهو الذى يتأكد من اذا كانت البيانات المرسلة صحيحة أم خطأ نتيجة تأثير noise اثناء ارسلها وتحدثنا عنه فى الدرس الاول فارجع اليه وبعدين ال stop bit وهى التى تحدد نهاية الارسال وبتخلى pin الميكرو بواحد عشان تتهيئ لاستقبال بيانات جديدة .

ولكن ليه ببعت الداتا بالاسكى كود ؟؟؟؟

دايما كنت بسال نفسى لية دايما اما كنت ببعت الداتا ل lcd كان لازم تبقى اسكى او ابعت للكمبيوتر او اى جهاز خارجى لحد ما فلويد حللى المشكلة دى كل جهاز بيبقى لية لغة بيفهمها مثلا انا بتكلم عربى حد تانى بيتكلم انجليزى وحد تالت بيتكلم لغة تانية بس فى النهاية علشان اتكلم مع اى حد لازم اتكلم بلغة كلنا نفهمها الى هيا الاسكى .
دلوقتى انا اما كنت ببعت الداتا للكمبيوتر كنت ببعتها اسكى والكمبيوتر رد عليا اسكى مثلا الكمبيوتر بعتلى رقم 1 فهيبعتهولى ب 49 انا دلوقتى الفكرة ان المتغير الى استقبلت فية الداتا متخزن فية قيمة 49 بس دى مش قيمة حقيقية علشان كده هطرح الرقم من 48 علشان يدينى قيمتة الحقيقية
انما بقى الداتا دى لو انا هبعتها لجهاز تانى اغير فى قيمتها لية مالجهاز التانى بيفهم اسكى فبعتلة الداتا علطول .



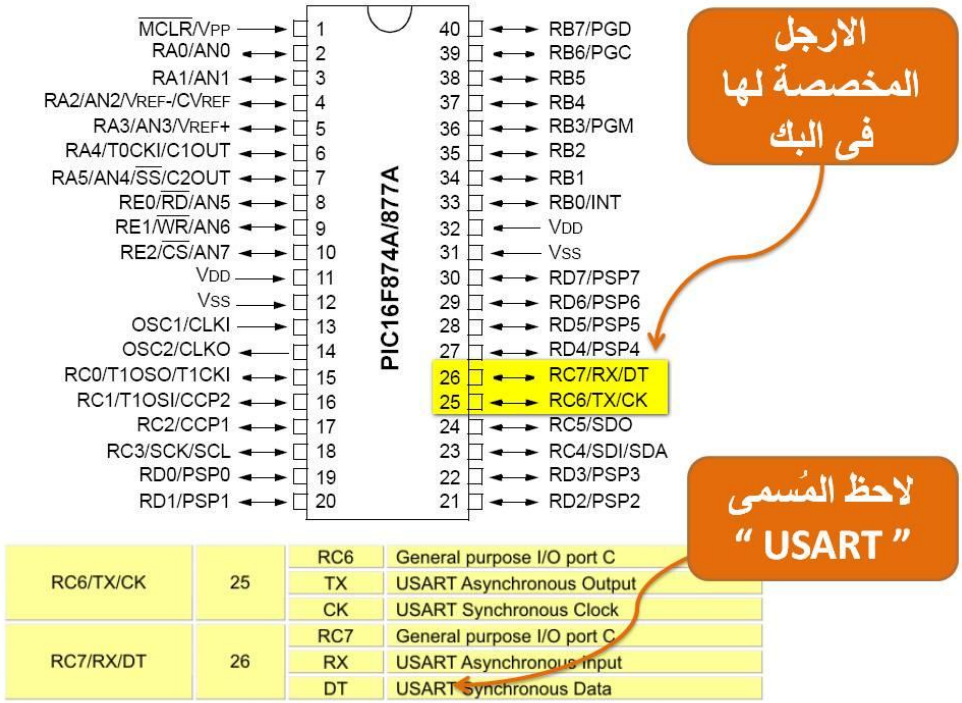
1. *When a serial port pin is idle (not transmitting data), it is in an on state.*
2. *When data is about to be transmitted, the serial port pin switches to an off state due to the start bit.*
3. *The serial port pin switches back to an on state due to the stop bit(s). This indicates the end of the byte.*

عندما تكون ال pin idle بمعنى أن مفيش داتا بتتنقل بيكون عليها logic high اول ما الداتا تتبعت بتصبح low بسبب ال start bit وبعد ما يتم ارسال الداتا تصبح high بسبب ال stop bit .

Bytes versus values

هو بيعتبر أن ال start bit وال data bits وال parity bit وال stop bit انهم byte ولكن هما كده 11 bits ومن المعروف أن ال byte عبارة عن 8 bits طب ليه بيعتبرهم انهم byte؟؟؟

بيعتبرهم byte عشان الميكرو بياخد ال data bits فقط .

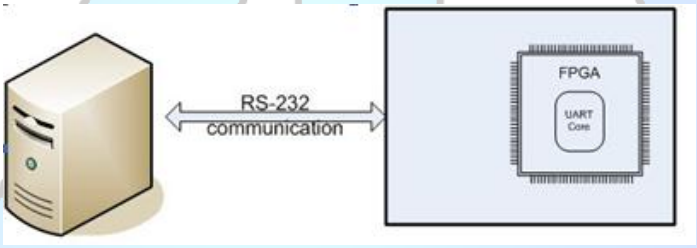


ماهو ال USART ؟؟؟؟

و هو عبارة عن شريحة صغيرة(IC)



من المعلوم أن الميكرو بيعمل على LOGIC LEVEL 5V/0V والسريال بيشتغل على logic level مختلف كما ذكرنا سابقا لذا لابد من دائرة hardware تقوم بتحويل ال logic level الخاص بالميكرو لل logic level الخاص بالميكرو والعكس عشان البيانات تنقل بصورة صحيحة .



مفاهيم أساسية Bit Rate & Baud Rate :

أتينا لجزء مهم في هذا النوع من الاتصال مسألة معدل النقل

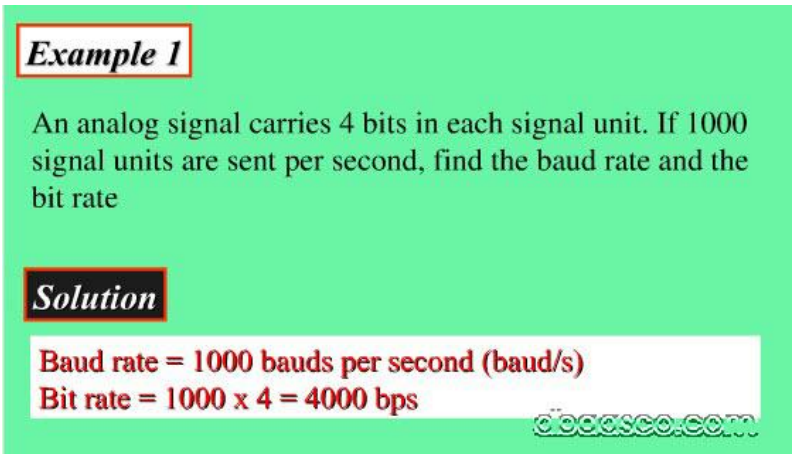
Bit Rate :

هو عبارة عن معدل نقل البتات في الثانية bps سواءاً كانت أصفار أو آحاد.

Baud Rate :

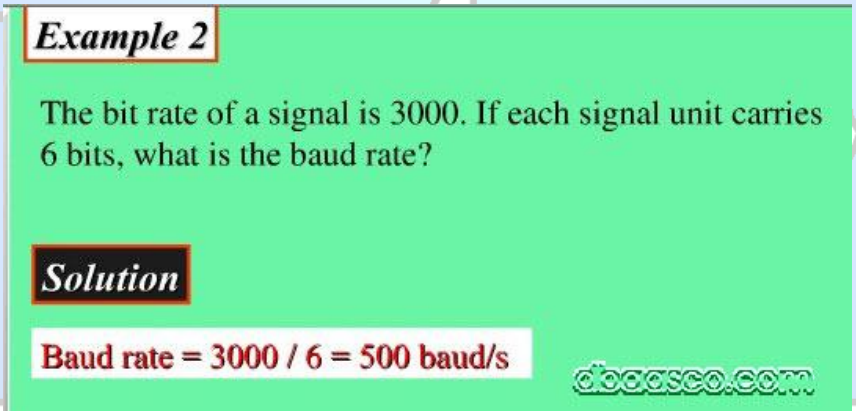
عدد الرموز في الثانية و قد تكون أكثر من بت في الثانية و يقاس بالتردد .

الـ Baud Rate قد يساوي Bit Rate عندما ينقل كل بت في الثانية شاهد الصور التالية و الأمثلة

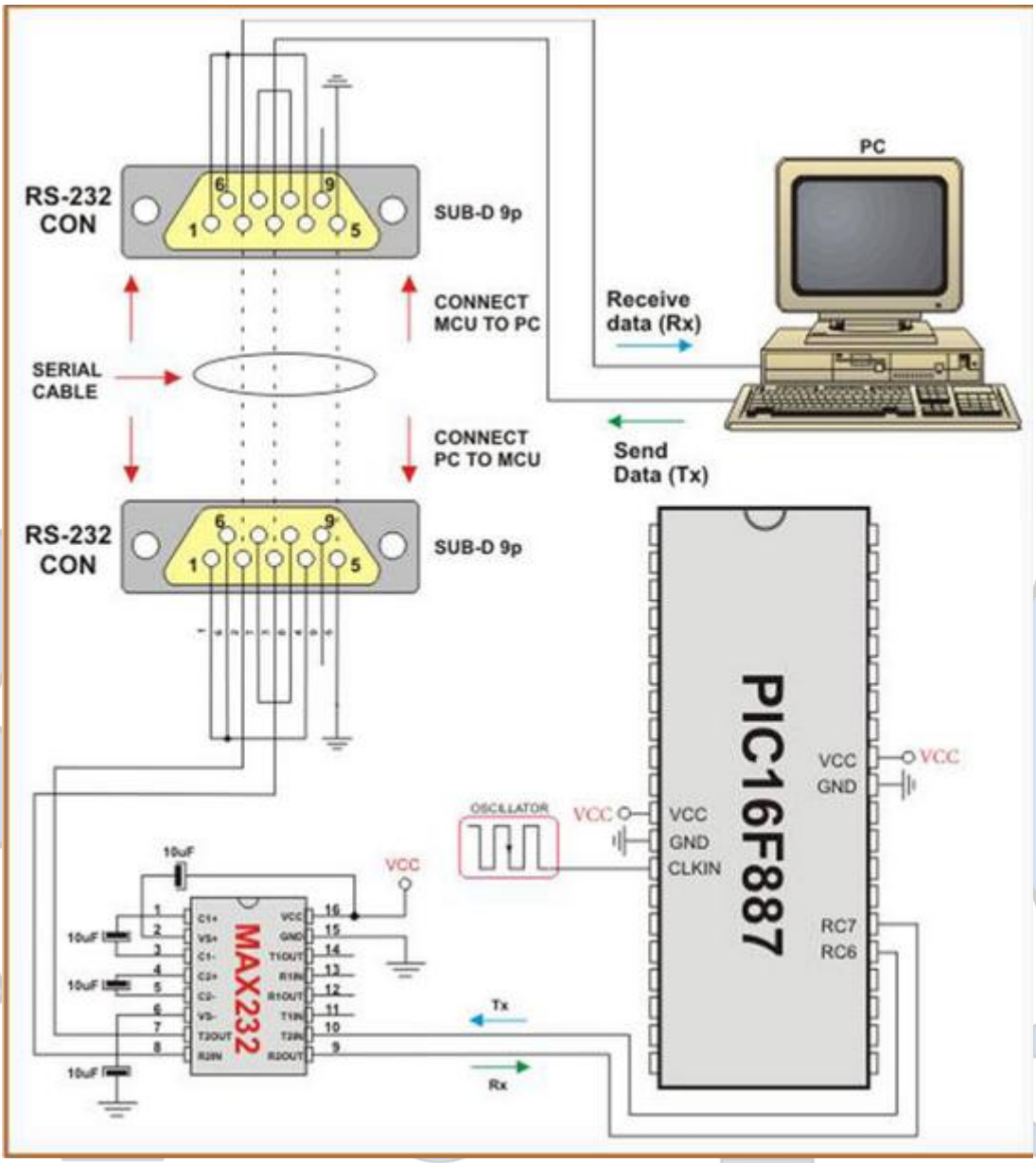


المثال واضح و هو أنه في كل إشارة حاملة تحمل معها 4 بتات فلو كان عندي 1000 إشارة حاملة كم معدل النقل ؟؟ الحل واضح

نأخذ مثال ثاني



كيفية التوصيل مع الكومبيوتر



فنجد أن pin 15 لل max232 موصلة بال GND و pin 16 موصلة بال VCC و pin 14/13/12/11 غير مستخدمة

ونجد أن pin 10 مكتوب عليها T2IN يعنى بيدخلها بيانات من الميكرو لذا توصل بال transmit للميكرو وهو pin RC6 وايضا pin 9 لل max232 مكتوب عليها T2OUT يعنى بيخرج بيانات منها للميكرو لذا نوصلها على ال receive للميكرو اللى هى pin RC7

نجد أن PIN7 لل MAX232 مكتوب عليها T2OUT يعنى بيخرج من ال MAX عشان الكومبيوتر يستقبل لذا يوصل ب PIN 2 فى السريال والمسئولة عن الاستقبال للبيانات وايضا PIN8 لل MAX مكتوب عليها IN لذا بيدخلها بيانات من السريال لذا توصل بب PIN3 فى السريال لانها المسئولة عن ارسال البيانات .

باقى ال PINS موصلة بمكثفات مع العلم ان الخط الاسود يمثل موجب المكثف والطرف الذى يشبه القاعدة يمثل الطرف السالب للمكثف .

الاوامر فى ال mikro C :

الأمر الأول

UARTX _Init (const unsigned long baud _rate);

Example	`// Initialize hardware UART1 and establish communication at 9600 bps` `UART1_Init(9600);`

ويمكن تغيير الرقم 9600 وهو سرعة الارسال وهو عدد البت فى الثانية فلو عندى حرف مكون من 8bit فانه بالاضافة الى ال start bit وال stop bit يصبح عددهم 10 bit يعنى 9600/10=960حرف يعنى يتم ارسال 960 حرف فى الثانية الرقم 9600 هو الرقم ال default .

الرقم X يشير الى رقم ال UART المستخدم لان هناك ميكرو لها اكتر من UART أما PIC16F877A لها UART واحد فقط لذا نكتب UART1 وان لم نكتب 1 سيحدث خطأ اثناء عملية ال COMPILING .

الأمر الثاني

UARTx _Data_Ready();

Example	`// If data is ready, read it:` `if (UART1_Data_Ready() == 1) {` `receive = UART1_Read();` `}`

هذا الأمر يختبر ما إذا كان هناك بيانات مرسلة من الجهاز للميكرونترولرهل الداتا الموجودة فى ال wire اتبعتت ولا لا وبيعرف انها اتبعتت من ال start bit أو بمعنى آخر هذا الأمر يحتمل قيمتين إما واحد في حال تم استقبال بيانات أو صفر فى حالة عدم استقبال بيانات .

طبعا يجب علينا أن نجعل البك يختبر باستمرار هل تم إرسال بيانات أم لا . كما في استخدام السويتشات . وذلك بأن نضع الأوامر السابقة في loop كما يلي :

While (1){

If (UARTx _ Data_ Ready()){

.....}

أيضا في لغة السي إذا أردنا أن نختبر شيء معين ونقول هل قيمته تساوي واحد فإننا يمكننا أن نكتب ذلك بطريقتين : الطريقة الأولى بأن نكتب علامتي تساوي والطريقة الثانية بدونهما

If (UARTx _ Data _Ready()==1){

فمثلا الأمر السابق يمكن كتابته كما يلي

If (UARTx _ Data _Ready()){

أي بدون 1== . فكلا الطريقتين لهما نفس الوظيفة .

الأمر الثالث

UARTx _ Read();

Example	

```
// If data is ready, read it:
if (UART1_Data_Ready() == 1) {
 receive = UART1_Read();
 }
```

هذا الأمر يستقبل بايت واحد فقط أي وظيفته استقبال بايت قادم من الجهاز للميكرونترولر في حين الأمر السابقUARTX _ Data_ Ready كان يختبر فقط وجود بيانات أم لا .

كلمة receive عبارة عن متغير فقط و يجب أن يكون هذا المتغير من نوع char و من خلال هذا المتغير اخزن البايت القادم من الجهاز عن طريق الأمر ;uartx_read() فى المتغير receive .

أي الآن استطيع مثلا لو ارسل حرف a او b أو أى حرف من حروف الكومبيوتر للميكرو أن أعرضه على شاشة LCD وهذا الامر يستقبل bit by bit .

عند التواصل بين الكمبيوتر والبك الانتباه إلى الحروف الكابتال (الكبير upper case)
والحرف الصغيرة lower case فكل حرف له كود مختلف .

الأمر الرابع

UARTx _Write(char data_);

Example	

```
unsigned char _data = 0x1E;
...
UART1_Write(_data);
```

هذا الامر يرسل بيت واحدة فقط من الميكرو للكومبيوتر اذا تم ارسال اكتر من بيت فانه يحدث error كما فى هذا الكود :

UART1_ Write("aaa")

الأمر الخامس

UARTx _Write_Text(char * UART_text);

Example

Read text until the sequence "OK" is received, and send back what's been received:

```
UART1_Init(4800);                       // initialize UART1 module
Delay_ms(100);

while (1) {
  if (UART1_Data_Ready() == 1) {        // if data is received
    UART1_Read_Text(output, "OK", 10);  // reads text until 'OK' is found
    UART1_Write_Text(output);           // sends back text
 }
}
```

هذا الامر لارسال text فلو أردنا ارسال كلمة engineer سنكتب الاتى

UART1_Write_Text("engineer");

هذا لو اردت ارسال كلمة علطول أما لو أردت ارسال متغير استخدم الامر الاتى

هذا هو المتغير

char output[]="Hi World!";

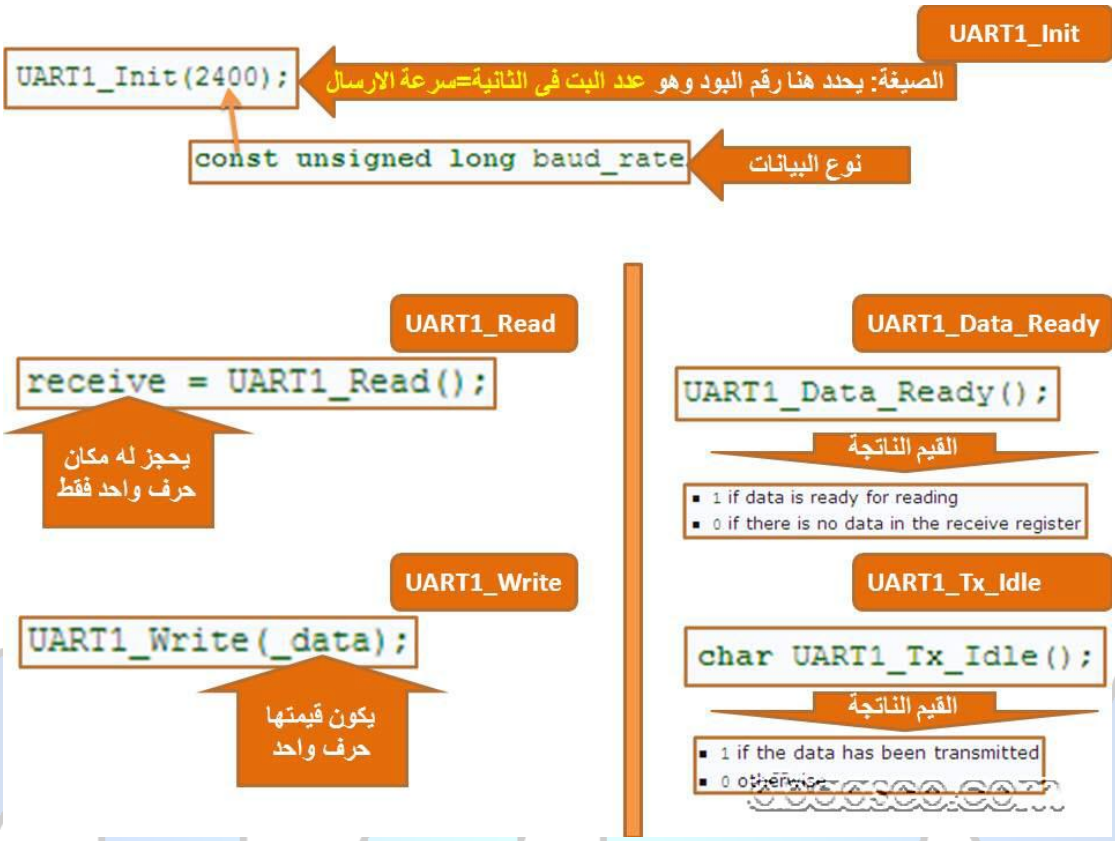
لاحظ الامر uart1_read_text(output,"ok",10) ال output هو المتغير اللى هيقرى منه ال text أما ok أنا هفضل ابعت داتا لحد ما يجيلة كلمة ok هيتوقف عن استقبال الداتا .

لارسال المتغير استخدم الامر :

UART1_Write_Text(output);

فى هذة الحالة لن يظهر كلمة output ولكن يرسل ما بداخل هذا المتغير على عكس المثال السابق فكان يعرض كلمة engineer علطول .

لاحظ أن الامر write اذا تم تخزين البيانات اللى فيه فى متغير فان المتغير يكون حرفى char زى الامر read بالضبط ويسع المتغير 50 حرف .



نقوم بعمل برنامج بسيط لتوضيح ما سبق

المشروع الاول

```
// Lcd pinout settings
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;

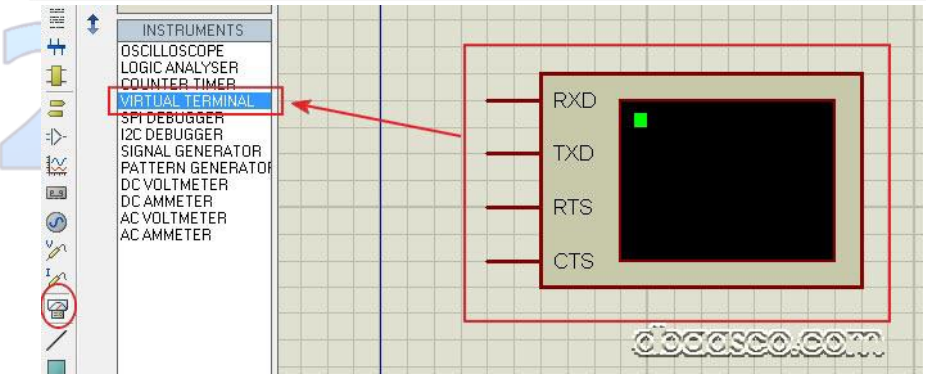
// Pin direction
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;
int i=0;
 char uart_rd[2];
void main() {
Lcd_Init();                        //initialize lcd
Lcd_Cmd(_LCD_CURSOR_OFF);          // Cursor off
 Lcd_Out(1, 1, "eng"); // Write text eng on Lcd starting from row 1, column 1:
  UART1_Init(9600);// Initialize hardware UART1 and establish communication at 9600 bps
  UART1_Write_Text("engineer 2012");             // sends  text
  while (1) {
  if (UART1_Data_Ready()) {          // if data is received
  uart_rd[0]= UART1_Read();     // reads text
       i++;
    UART1_Write(uart_rd[0]);             // sends back text
    lcd_out (2,i,uart_rd);
 }
}
}
```

يقوم البرنامج بعرض البيانات المرسله من الكومبيوتر على شاشة ال LCD ففى البداية نقوم بتعريف ال lcd ثم ندخل فى دالة main ونعمل initialize لل lcd

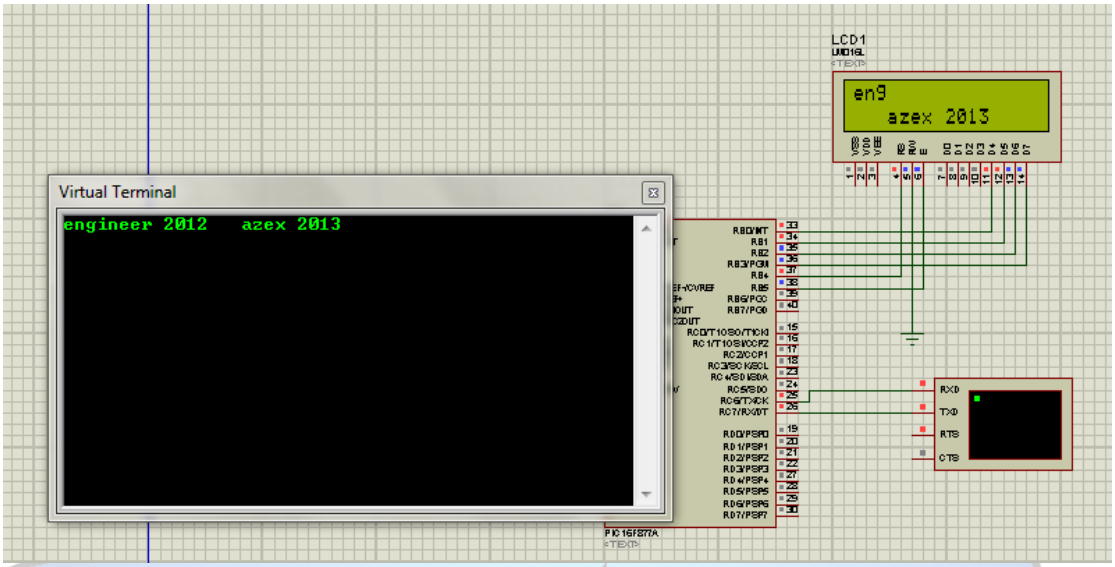
وبعدين نعمل initialize لل uart ثم نعرض كلة eng على ال lcd فى السطر الاول وذلك للتأكد من أن البرنامج شغال فعندما اشغل البرنامج لو لقيت كلمة eng مكتوبه يبقى البرنامج شغال وبعدين هكتب على شاشة الكومبيوتر engineer 2012 باستخدام الامر write text وبكده اول ما اشغل البرنامج هيظهر كلمة eng على ال lcd ويظهر كلمة engineer 2012 على الكومبيوتر وبكده ندخل للبرنامج الاساسى داخل while لازم داخل while أتاكد ان البيانات جاهزة عن طريق الامر if (UART_DATA_READY()){ وبعدين أقرى البيانات اللى مرسلة من الكومبيوتر واخزنها فى المتغير uart_rd واللى لابد ان يكون من النوع char البيانات اللى الكومبيوتر هيرسلها هطلب من الكومبيوتر انه يعرضها باستخدام الامر write وفى نفس الوقت يعرضها على ال lcd فى السطر الثانى فلاحظ فى البداية هيظهر eng و engineer 2012 كما ذكرنا مسبقا وأيضا لو كتبت ع الكومبيوتر حرف فان هذا الحرف سيظهر ع الكومبيوتر وايضا ع ال lcd فى السطر التانى العمود الاول ثم i تزيد بمقدار واحد فلو كتبت حرف تانى هيتكتب ع الكومبيوتر وهيظهر ع ال lcd فى السطر التانى العمود التانى بجوار الحرف الاول وهكذا

لكن ازاى ابعت داتا من الميكرو للكومبيوتر والعكس عن طريق البروتس ؟؟؟

الاداة virtual terminal هى التى عن طريقها نتاكد ان البرنامج شغال فى هى تضم دائرة ال hardware والتى تتحكم فى ال logic level وهى دائرة ال max232 وايضا تضم الكومبيوتر وناتى بها من ال virtual instruments mode على الجانب الايسر من البروتس ولها طرفين طرف استقبال RXD ويوصل بطرف الارسال RC6 فى الميكرو وايضا له طرف ارسال TXD ويوصل بطرف الاستقبال RC7 فى الميكرو .



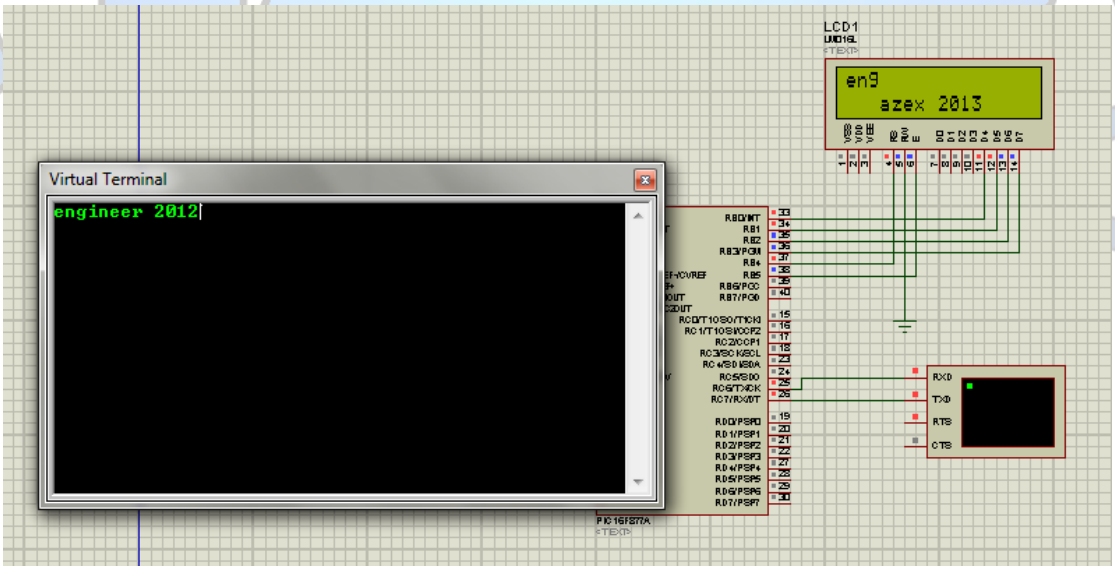
توصيل الدائرة :



ستجد انه عند كتابتك كلمة azex 2013 فى الكومبيوتر فانه يتم كتابتها فى الكومبيوتر والذى يمثل الشاشه السمراء ويتم عرضها ع الشاشة

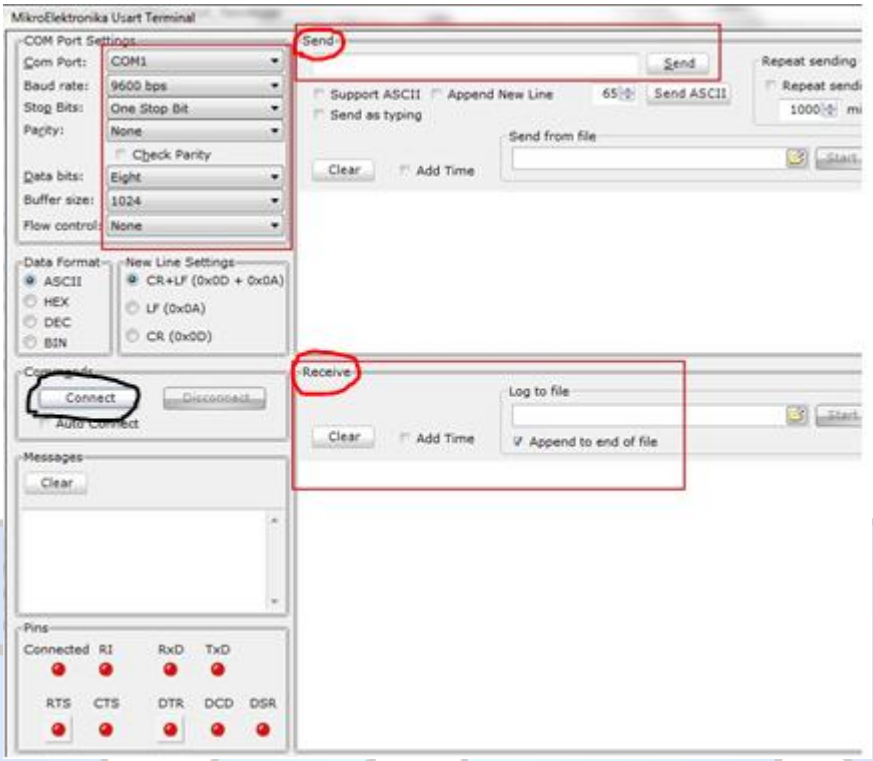
لكن كيف أتاكد أن الكلمة azex 2013 اللى أرسلتها من الكومبيوتر للميكرو فعلا اتكتبت على الكومبيوتر مش أنا اللى كتبتها ؟؟؟؟

لو شلنا الامر السابق من الكود `UART1_Write(uart_rd[0]);` هنجد أنى بكتب ع الكومبيوتر بيظهر ع الشاشة كلمة azex 2013 ومش بيظهر ع الكومبيوتر كما هو موضح بالصورة



طب فى الحقيقة هبعت ازاى من الكومبيوتر للميكرو ايه ال graphical user interface اللى هبعت بيها أو هستقبل عليها هل notepad ولا شئ اخر ؟؟؟

هاستخدم tool موجودة فى الميكروسى من قائمة tool هختار UART terminal هتظهر القائمة التالية :



هختار من com port هختار com 1 وهختار من ال band rate الرقم اللى كتبته فى البرنامج وهو 9600 وبعد ما وصلت كل حاجة برمجة الميكرو ووصلت دائرة ال max عن طريق كابل السريال بالكومبيوتر ووصلت دائرة ال max بالميكرو هضغط connect كما فى الصورة اللى عاوز ابعته اكتبه فى الخانة send وأقوله send واللى الكومبيوتر هيستقبله هيستقبله فى الخانة receive .

المشروع الثانى

فكرة البرنامج :

يقوم الميكرو باستقبال بيانات من الكومبيوتر واعادة ارسالها الى الكومبيوتر

الكود :

```
char uart_rd;
void main() {
  UART1_Init(9600);               // Initialize UART module at 9600 bps
  delay_ms(100);                  // Wait for UART module to stabilize
  UART1_Write_Text("Start");
  UART1_Write(10);
  UART1_Write(13);
  while (1) {                     // Endless loop
  if (UART1_Data_Ready()) {      // If data is received,
     uart_rd = UART1_Read();      // read the received data,
   UART1_Write(uart_rd);        // and send data via UART
  }
  }
}
```

فى البداية سيظهر ع الكومبيوتر كلمة start من خلال الامر

UART1_Write_Text("Start");

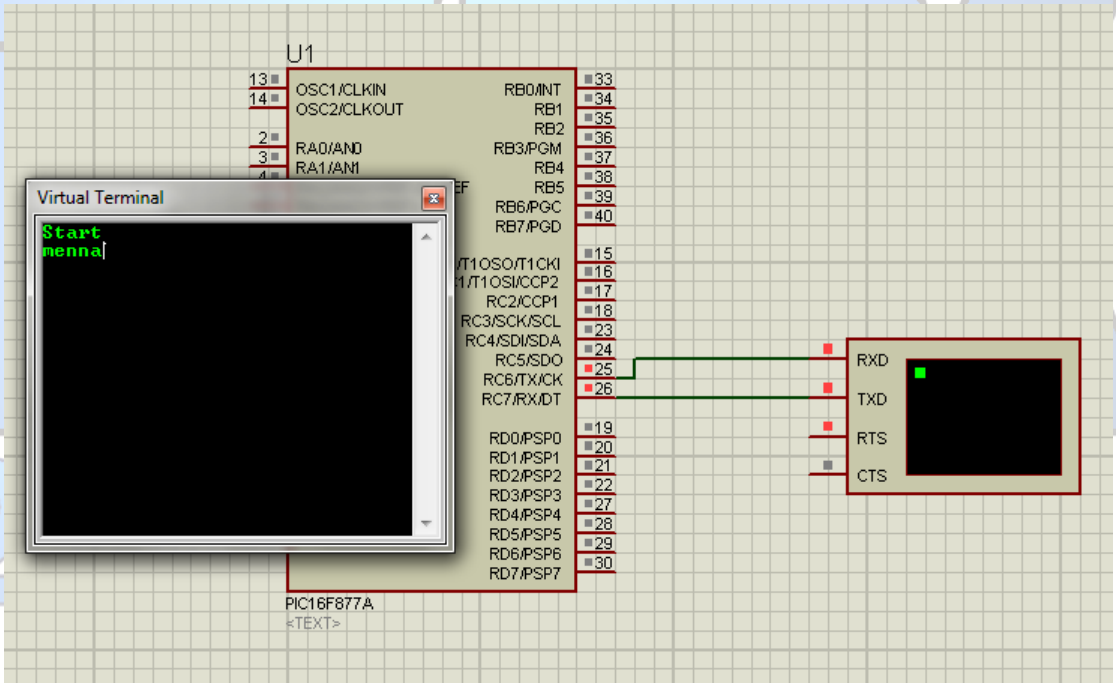
كذلك قلت له اجعل المؤشر ينتقل لسطر جديد من خلال الأمر التالي

UART1_Write(10);

UART1_Write(13);

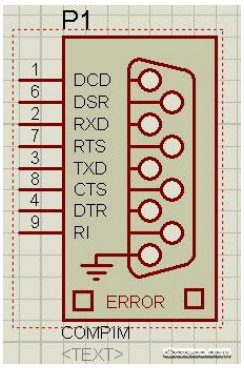
أو ما يسمى بلغة السى \n سطر جديد .

توصيل الدائرة :

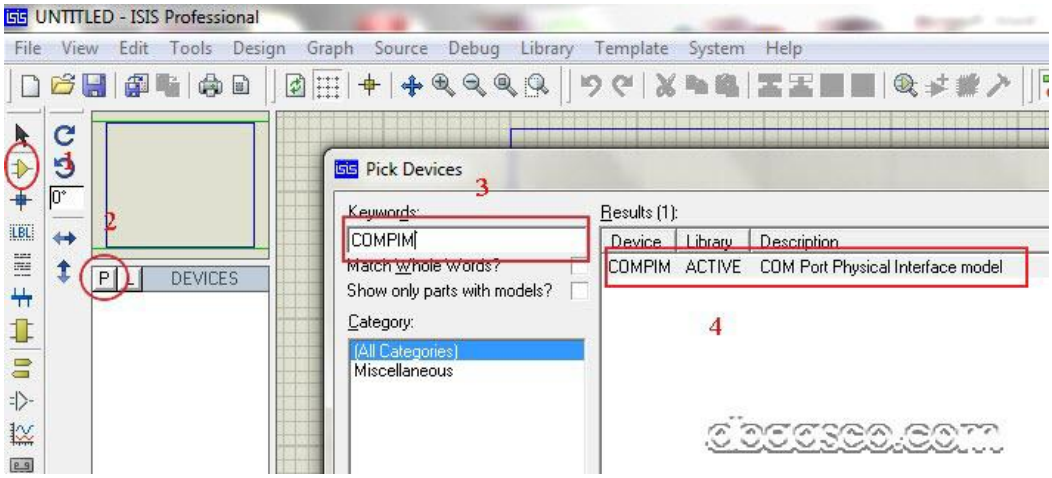


قد يسأل المبرمج اين ال RS232؟؟؟؟

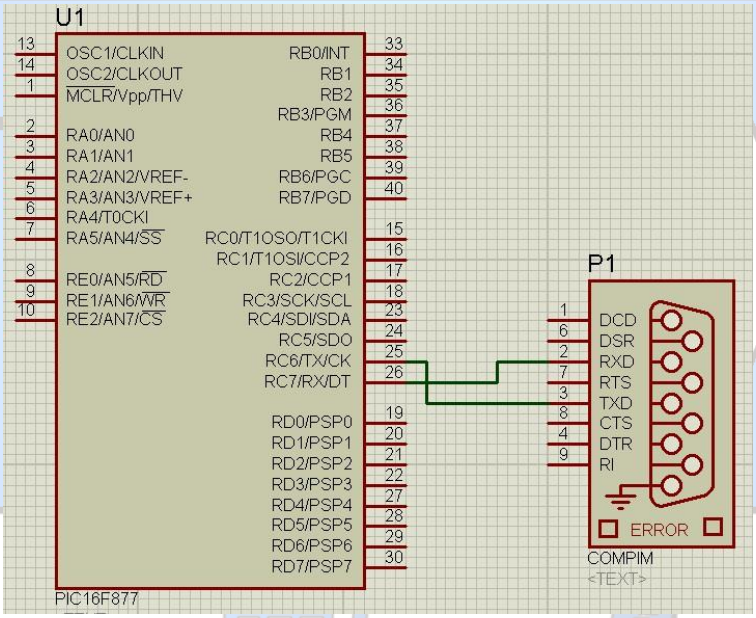
الميكرو سى يوفر لك الاداة virtual terminal وايضا ربط الميكرو بالكومبيوتر عن طريق ال RS232 والتى تكون فى بروتس بهذا الشكل



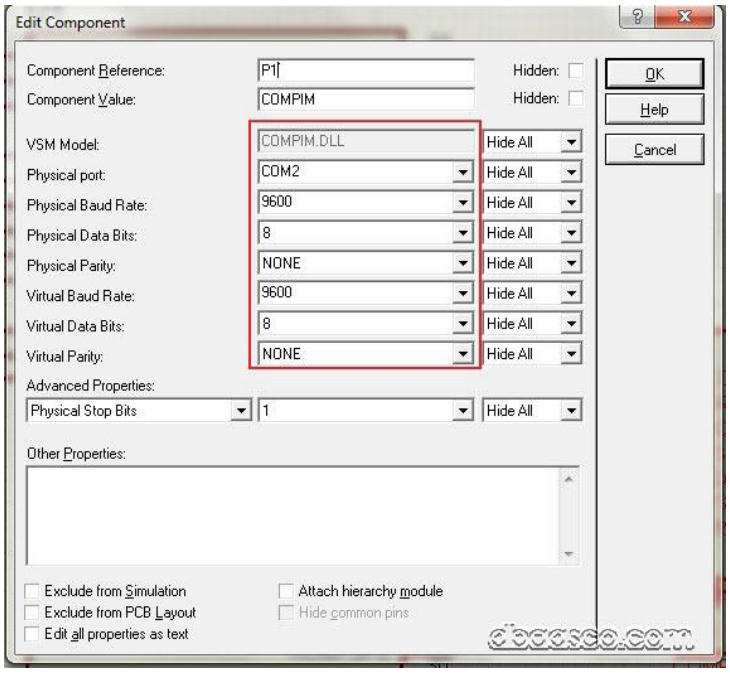
ويكون اسمها فى البروتس COMPIM



يتم توصيلها كالاتى :



اضغط على ال RS232 ضغطتين بالماوس ستظهر لك شاشة قم بتضبيط الاعدادات التى بها كما فى الصورة :

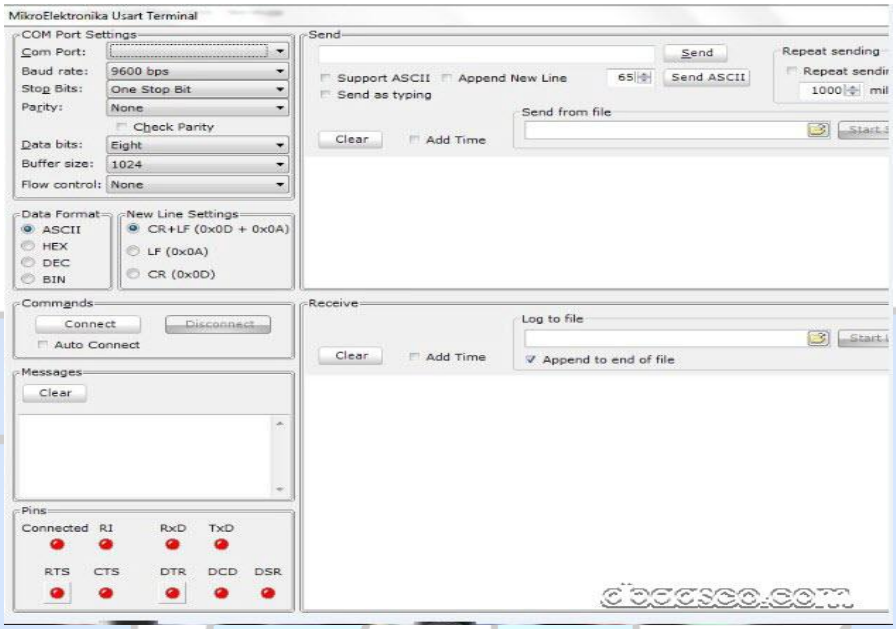


لاحظ أختارنا للبروتس com2 لاننا أخترنا للميكرو سى com1 .

ويتم الارسال والاستقبال من الاداة الموجودة فى الميكرو سى

Tools>>USART Terminal

والتى يتم من خلالها التاكد من الكود والارسال والاستقبال من خلال الاداة RS232



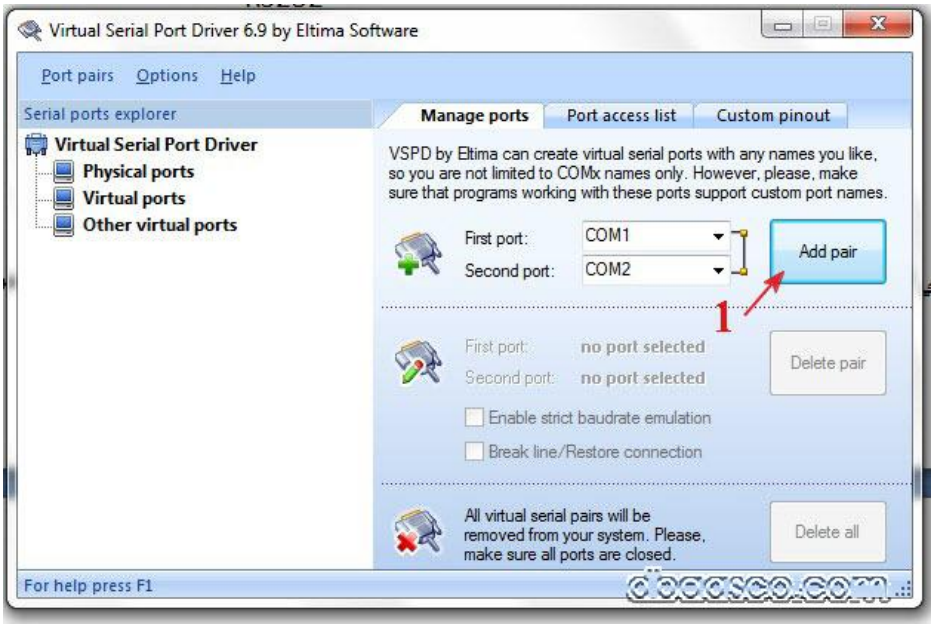
وهذة الاداة تحتاج الى منفذين COM1 & COM2 واحد للميكروسى والتانى لل RS232 ع البروتس

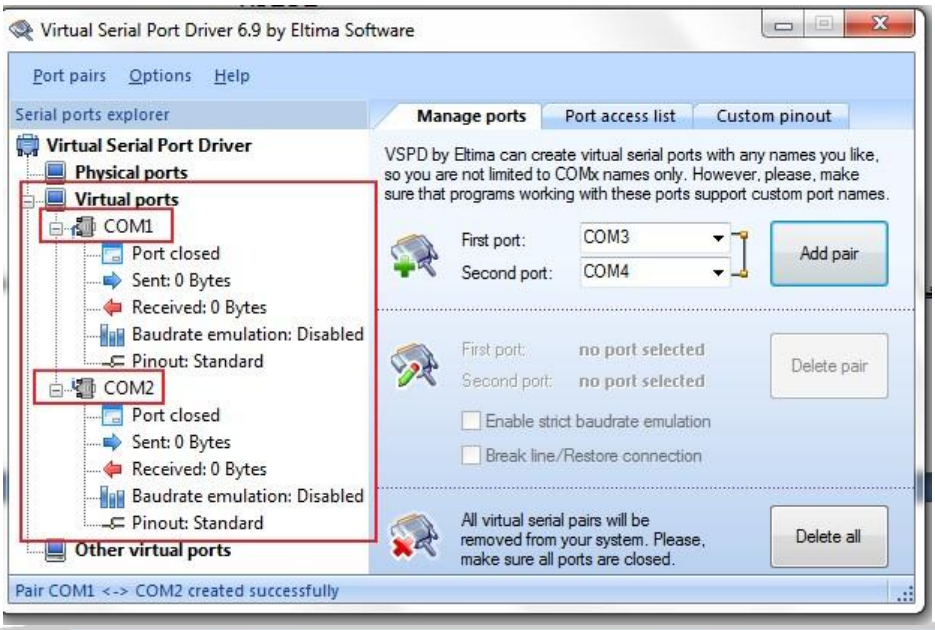
البرنامج الذى يقوم بانشاء المنفذين هو

Virtual Serial Ports Driver 6.9

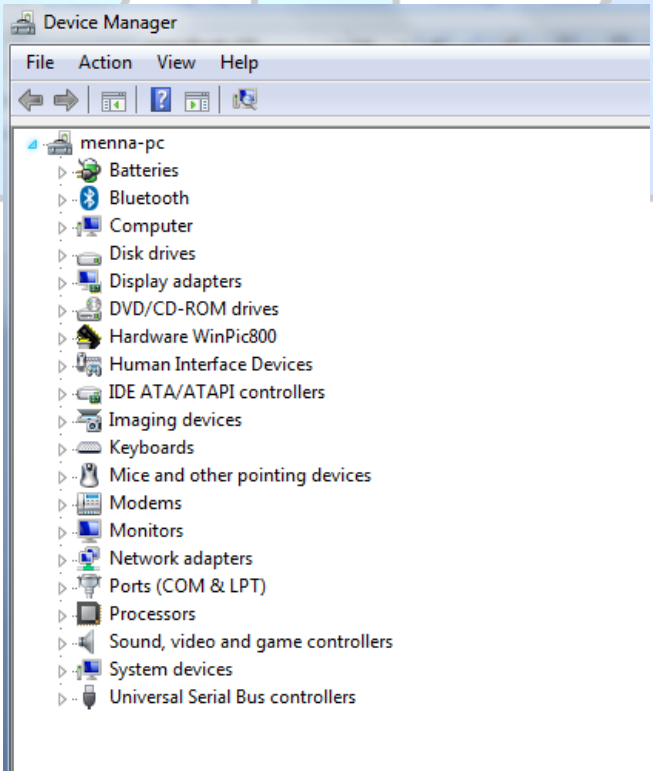
نقوم بتسطيبة الى ان تظهر الصورة التالية نضغط على add pair

سيظهر الاتى

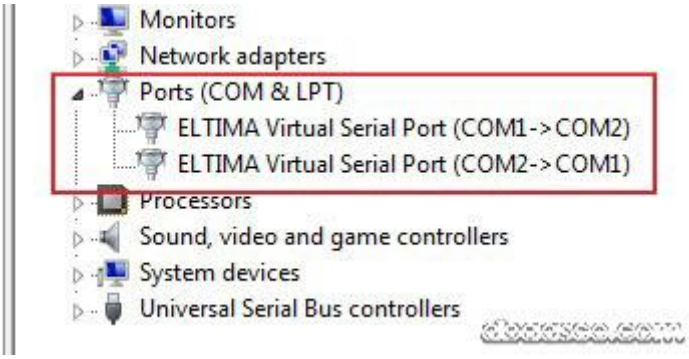




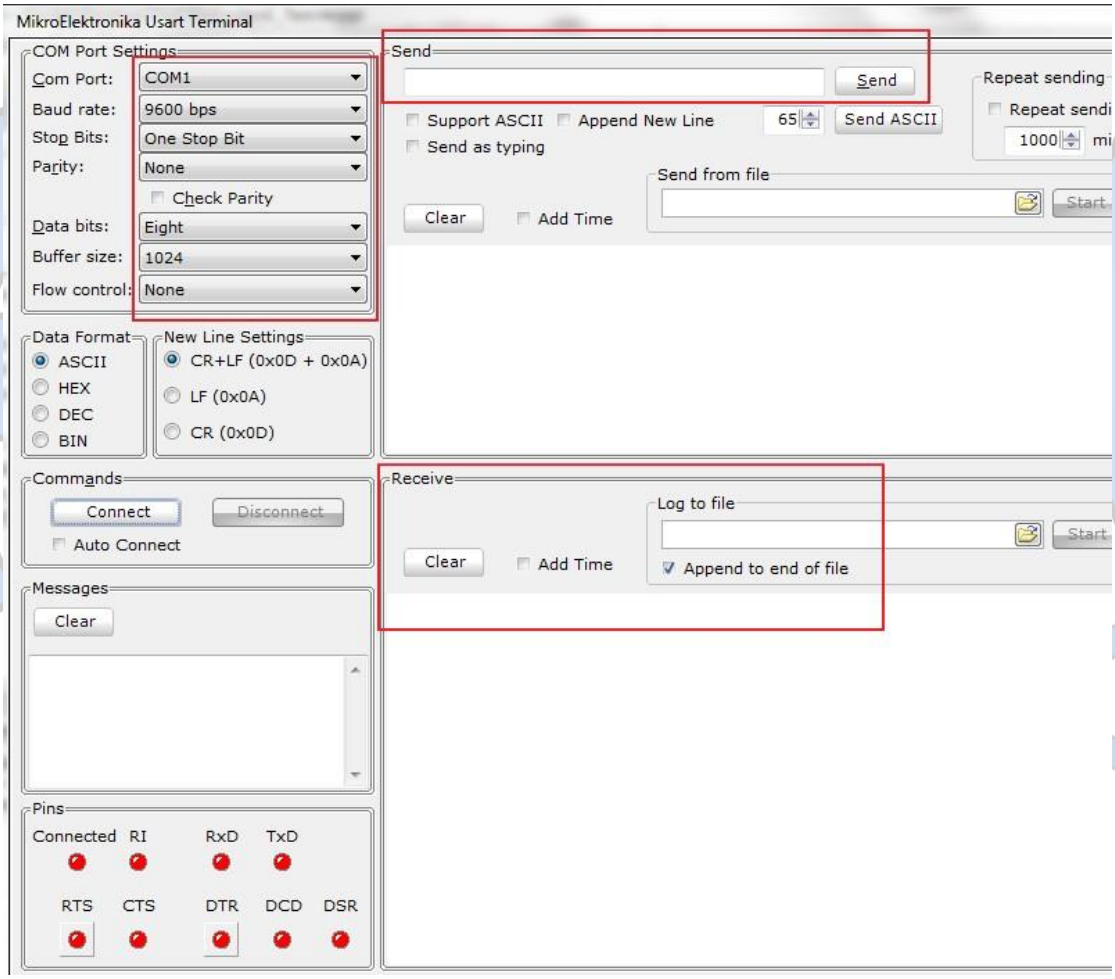
للـتأكد من وجود المنفذين الذهاب لمنافذ الجهاز عن طريق ضغط click يمين بالماوس على my computer هتظهرقائمة اختار منها properties ستظهر صفحة اختار منها device manger الموجودة على يسار الصفحة ستظهر لك صفحة اخرى بهذا الشكل :



اختار منها ports وتاكد من وجود الموجود بالصورة التاليه :



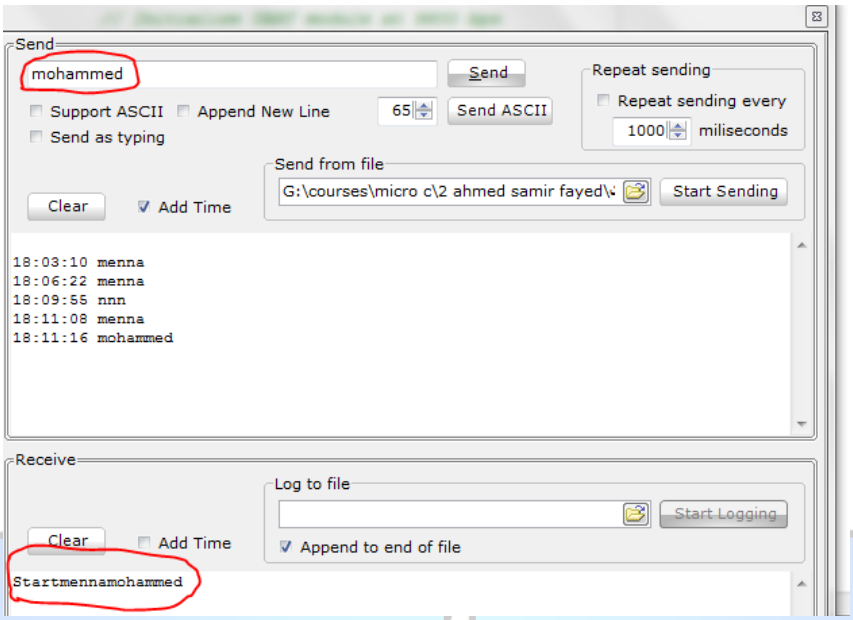
بعد ذلك نذهب للميكروسي لضبط الإعدادات ونختار ليه port معين وليكن com1



اذهب للبروتس واضغط run وشاهد عملية الارسال والاستقبال

لاحظ لا تضغط run فى البروتس قبل أن تضغط connect فى أداة الميكرو سى

عند تنفيذ البرنامج البرنامج التانى يظهر ع الميكرو سى :



المشروع الثالث

فكرة البرنامج :

برنامج عندما يرسل الكومبيوتر للميكرو حرف ال g يقوم باضاءة ليد

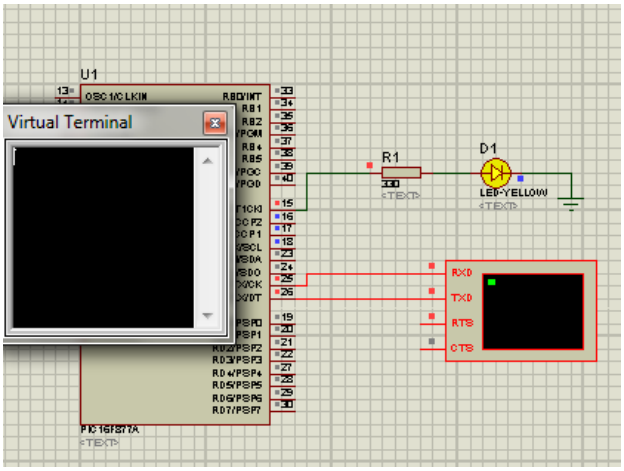
الكود :

```
char recieve;
void main() {
 trisc=0b11110000;   //portc is output
  portc.f0=0;
  UART1_Init(9600);// Initialize hardware UART1 and establish communication at 9600 bps


  while (1) {
  if (UART1_Data_Ready()==1) {          // if data is received
  recieve= UART1_Read();    // reads text
if (recieve=='G')
portc.f0=1;
}
}
}
```

لاحظ pin RC0 خرج هنضع عليها الليد و PIN RC7 دخل لانها تستقبل من الكومبيوتر لذا يجب تعرفها فى TRIS بانها 1 يعنى دخل .

توصيل الدائرة :



البرنامج الرابع

فكرة البرنامج :

يقوم ال user بادخال كلمة سر az_ex الى الكومبيوتر فاذا ادخلها يقوم الكومبيوتر باظهار رسالة لل user مكتوب فيها welcome azex orgnizer yes واذا لم يدخل الاسم الصحيح يظهر الكومبيوتر رساله مكتوب فيها wrong enter right name .

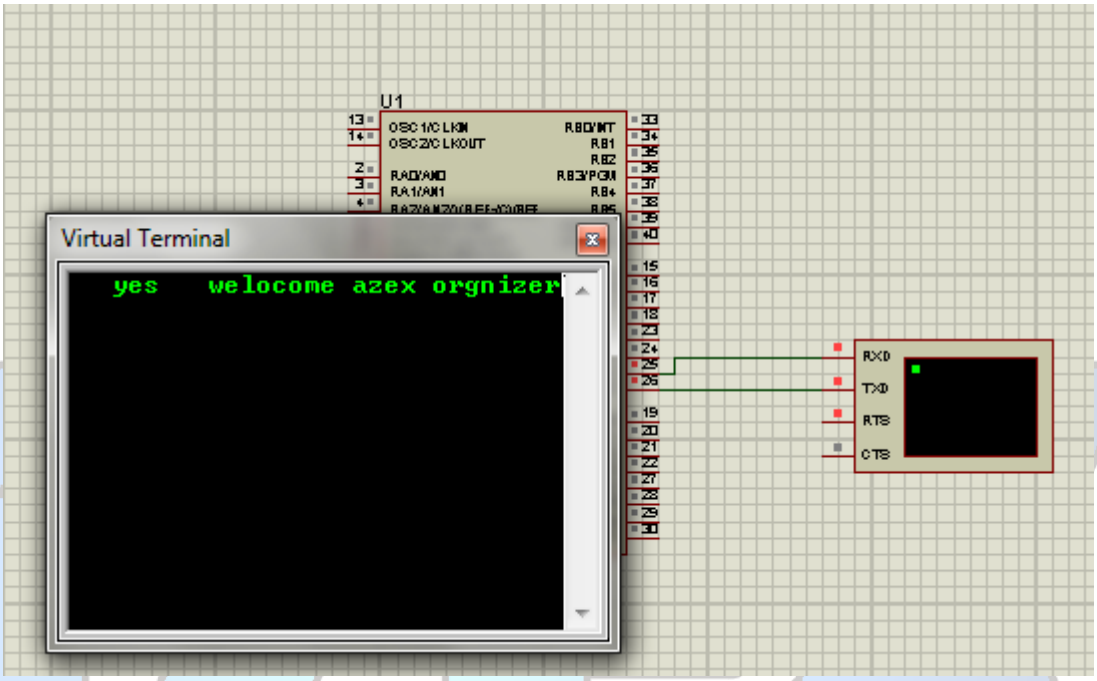
الكود :

```
char x[5];
char y=0;
void main() {
  UART1_Init(9600);// Initialize hardware UART1 and establish communication at 9600 bps
  while (1) {
  if (UART1_Data_Ready()==1) {          // if data is received
  x[y]= UART1_Read();    // reads text
  y++;
if (y==5){
if (x[0]=='a' &&x[1]=='z' &&x[2]=='_' && x[3]=='e' && x[4]=='x'){
  UART1_Write_Text("   yes   welocome azex orgnizer");
y=0;
}
else {
   UART1_Write_Text("   wrong  enter right name");
    y=0;
}
}
}
}
}
```

نقوم بتعريف متغير x مكون من 5 حروف عدد حروف كلمة az_ex فى البداية يقرأ الميكرو الحروف المدخله عن طريق الكومبيوتر والتى تدخل حرف يليه حرف لذا نقوم بزيادة قيمة المتغير y بمقدار واحد الى ان يتم ادخال ال 5 حروف فاذا ساوت y==5 فيقوم الميكرو باختبار الكلمة فاذا كانت az_ex يبعث برساله للكومبيوتر yes وتركنا مسافة قبل yes حتى لا يكون الكلام متشابك على شاشة الكومبيوتر ولكن اذا لم يدخل ال user كلمة az_ex يتحقق

شرط else ويظهر رسالة wrong يطلب من المستخدم ادخال الاسم الصحيح وهكذا لاحظ قمنا بكتابة y=0; وذلك لان y=5 فاذا لم نكتب هذا الامر فلا يستطيع المستخدم ادخال الاسم مرة اخرى .

توصيل الدائرة :



ربط البيكات ببعضها

يمكن أن ترسل بيك معلومات لبيك أخرى وتستقبلها هذة البيك , عن طريق توصيل طرف ال RX للبيك الأولى بطرف ال TX للبيك الثانية والعكس .

أول ما اضغط ع السوتش فى دايرة ال transmitter يبعت رقم 17 فى دايرة ال receive

عندى برنامجين واحد للبيك الأولى والاخر للبيك الثانية .

اذا كان عملية الارسال والاستقبال عن طريق 2 بيك ممكن أستخدم أى baud rate لكن ال transmit وال receive لابد أن يكون لهم نفس ال baud rate .

لكن فى الارسال والاستقبال عن طريق ال serial يفضل استخدام ال 9600 .

((راجع برنامج serial communication فى البرامج الملحقة))



AZ-EX 2013

Technical Support Committee